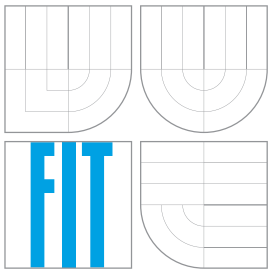
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ROZPOZNÁVÁNÍ AKTIVIT Z TRAJEKTORIÍ POHYBUJÍCÍCH SE OBJEKTŮ

ACTIVITY RECOGNITION FROM MOVING OBJECT TRAJECTORIES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. IVAN SCHWARZ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MARTIN PEŠEK

BRNO 2013

Abstrakt

Cílem diplomové práce je vytvoření systému pro rozpoznávání periodických vzorů pohybujících se objektů a následná klasifikace uživatelských GPS trajektorií s využitím rozpoznaných vzorů. Systém je navržen na základě provedeného rozboru technik dolování v datech pohybujících se objektů a přehledu dosavadního vývoje v oblasti rozpoznávání aktivit a cílů pohybujících se objektů. Je vytvořena implementace navrženého systému v programovacím jazyce C++ a provedeno experimentální ověření jeho úspěšnosti na vhodné datové sadě.

Abstract

The aim of this thesis is a development of a system for trajectory-based periodic pattern recognition and following GPS trajectory classification. This system is designed according to a performed analysis of techniques of data mining in moving object data and furthermore, on recent research on a subject of a trajectory-based activity recognition. This system is implemented in C++ programming language and experiments addresing its effectiveness are performed.

Klíčová slova

GPS, trajektorie, dolování v datech, rozpoznávání aktivit a cílů, periodický vzor, klasifikace

Keywords

GPS, trajectory, data mining, activity recognition, periodic pattern, classification

Citace

Ivan Schwarz: Rozpoznávání aktivit z trajektorií pohybujících se objektů, diplomová práce, Brno, FIT VUT v Brně, 2013

Rozpoznávání aktivit z trajektorií pohybujících se objektů

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Martina Peška. V práci jsem uvedl všechny publikace a prameny, z kterých jsem čerpal.

.....
Ivan Schwarz
23. května 2013

Poděkování

Děkuji Ing. Martinu Peškovi za svědomité vedení při práci na diplomové práci a dále pak Ing. Michalu Kováčikovi za plodné diskuze nad tématem dolování v datech, které často trvaly až dlouho do noci.

Obsah

Kapitola 1

Úvod

Technologie pro sběr dat se neustále rozvíjejí a stávají se dostupnější pro širokou veřejnost. Měření je rychlejší a podrobnější. Objem získaných dat tak rapidně roste a jejich ruční zpracování je náročnější, nákladnější a v některých případech dokonce přestává být v lidských silách. Zvyšuje se tak potřeba vývoje automatizovaných prostředků pro zpracování těchto dat.

Systémy pro zjišťování polohy pohybujících se objektů tvoří velice plodnou a zajímavou podmnožinu těchto technologií. Senzory zachycující přírodní úkazy, GPS přijímače umístěné na těle zvířete či uvnitř auta nebo mobilního zařízení nám umožňují shromažďovat velké množství polohových dat téměř jakýchkoliv pohybujících se objektů. Analýza těchto dat nám může pomoci např. získat nové informace o zvycích zvířat, lépe řídit dopravu, dříve zasáhnout v případě kriminálních aktivit či přesněji předvídat příchod živelných katastrof. Extrakcí užitečných informací z polohových dat se bude zabývat druhý kapitola této práce.

Jak bylo zmíněno v předchozím odstavci, získané znalosti mohou nalézt uplatnění v mnoha návazných systémech. Zejména pak v úlohách klasifikačních a predikčních. Jedna z velkých oblastí zájmu je možnost rozpoznávání aktivit a cílů pohybujících se objektů. To může vést k lepší komunikaci mezi strojem a člověkem, možnému neagresivnímu monitorování starších či mentálně postižených lidí či zaznamenávání provedených aktivit a personalisovanému návrhu dalších. Třetí kapitola přináší přehled dosavadního vývoje v oblasti rozpoznávání aktivit a cílů z trajektorií pohybujících se objektů.

Čtvrtá kapitola obsahuje návrh vlastního systému pro rozpoznávání aktivit a cílů, konkrétně detekci periodického chování lidských objektů při zvolené periodě a následnou klasifikaci nově přijatých uživatelských trajektorií do vytvořených periodických vzorů. Návrh je založen na technikách popsaných v první kapitole a částečně inspirován některými z metod přístupů uvedených v kapitole druhé.

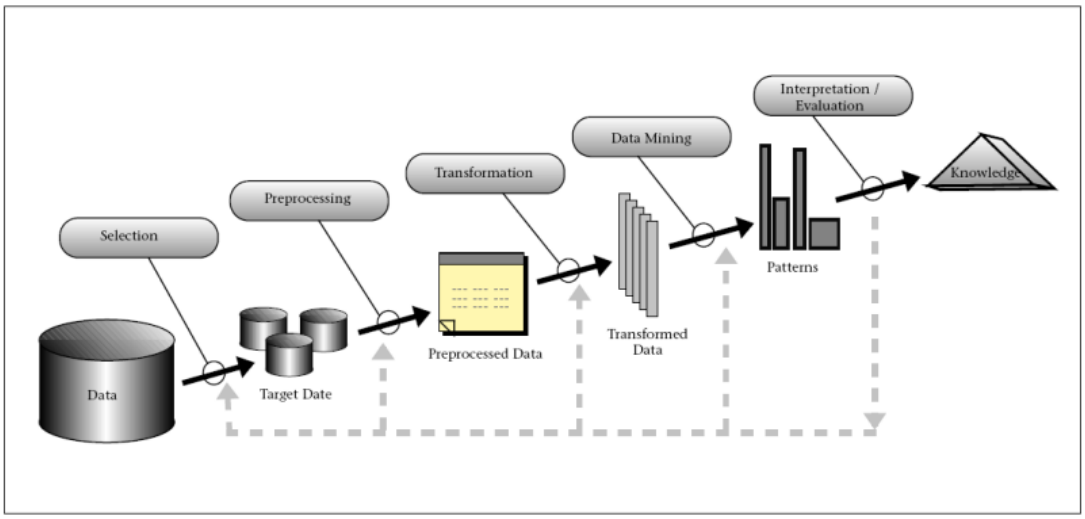
Navržený systém byl implementován v programovacím jazyce C++. Pátá kapitola se věnuje popisu implementace a použitým nástrojům a knihovnám. Je představen zvolený přístup k dekompozici problému. Část této kapitoly je také věnována vizualizaci polohových dat. Úspěšnost navrženého systému byla experimentálně ověřena na vhodné datové sadě. Výsledky těchto experimentů spolu s analýzou vlivu parametrů některých implementovaných metod a algoritmů jsou diskutovány v poslední, šesté kapitole.

V závěru je shrnuta celá práce, je diskutována úspěšnost zvoleného přístupu a nastíněna možnost budoucího vývoje.

Kapitola 2

Proces získávání znalostí z dat pohybujících se objektů

Pokud se budeme držet klasického schématu navrženého v [10], můžeme proces získávání znalostí z databází (angl. KDD – Knowledge Discovery in Databases) rozdělit na několik fází tak, jak vidíme na obrázku 2.1.



Obrázek 2.1: Schéma procesu získávání znalostí z databází podle [10].

Tato kapitola se věnuje popisu jednotlivých fází KDD procesu v souvislosti s dolováním v polohových datech. Nejprve popíšeme způsoby, kterými se tato data shromažďují. Následně vysvětlíme, proč je vhodné je před samotnou dolovací fází určitými způsoby předzpracovat. Popíšeme metody dolování v polohových datech a na závěr si nastíníme, jak z nich získané znalosti vyhodnotit a dále využít.

2.1 Polohovací systémy a sběr dat

Polohovacích systémů, které periodicky nebo na vyžádání určují polohu objektů, je široká škála. Většina z nich navíc zároveň tato data ukládá a shromažďuje. Ačkoliv jsou systémy

pro sběr polohových dat postavené na stejném či velmi podobném principu, lze je rozdělit do několika kategorií.

Tato sekce je ve velké míře inspirovaná dizertační prací [17], která obsahuje podrobný přehled a porovnání lokalizačních systémů.

Meziplanetární polohovací systémy

K určení polohy pohybujícího se kosmického objektu lze použít meziplanetární radio za předpokladu, že je tento objekt vybaven transponderem, který po přijetí radiosignálu ho vysílá zpět. Orientační informaci pak můžeme získat pomocí zařízení na určení polohy hvězd.

Globální družicové polohovací systémy (GNSS)

GNSS jsou systémy, které pomocí družic a radiových přijímačů dokáží určit polohu (zeměpisnou délku, zeměpisnou šířku a nadmořskou výšku) objektu v určitém čase s přesností na několik metrů. Některé z těchto systémů (GPS, GLONASS) mají celosvětové pokrytí. GNSS systémy lze kvůli atenuaci a odrazům signálu spolehlivě použít pouze k určení polohy objektu pod širým nebem. Kromě již zmíněných jsou dalšími příklady GNSS systémů projekty COMPASS a Galileo.

Zjišťování polohy uvnitř budov

Vzhledem k neúčinnosti GNSS uvnitř budov jsou v tomto směru vyvíjeny alternativní metody. Namísto využití satelitů se zaměřují na použití bližších objektů se známou polohou, zejména pak bezdrátových technologií (např. systém RADAR založený na WiFi signálech [4]).

Zjišťování polohy v rámci pracovního prostoru

Tyto systémy se zaměřují na pokrytí pouze malého, přesně vymezeného prostoru, nicméně poskytují přesnost v řádu milimetrů. Sem můžeme zahrnout techniky pro snímání pohybu, prostředí pro virtuální realitu atd. Speciální podskupinu pak představují systémy pro určování polohy objektů ve videu.

2.1.1 Dostupné datové sady

V současnosti je stále ještě obtížné získat ideální datovou sadu (dataset), která by dostatečně popisovala reálný svět z hlediska pohybu objektů a umožnila uspokojivé otestování vyvíjených algoritmů. Zejména tvorba datasetů poloh živých objektů stále probíhá uměle a v určitých ohledech mohou být tyto datové sady omezující (počet objektů apod.). Příklady některých takto vytvořených datasetů si uvedeme.

GeoLife GPS Trajectories

Tato datová sada byla v rámci projektu GeoLife[1] nasbírána 182 uživateli v průběhu 5 let (duben 2007 až srpen 2012). GPS záznamy každého účastníka jsou uspořádány do trajektorií. Dataset obsahuje 17621 trajektorií o celkové vzdálenosti 1292951 km a celkové délce 50176 hodin a je volně dostupný k nekomerčnímu použití, viz [45].

[1] http://research.microsoft.com/en-us/projects/geolife/

Movebank

Volně dostupná online databanka [39] obsahující velké množství datasetů polohových dat pohybujících se zvířat a dále pak studií na těchto datech založených. Ke stopování zvířat jsou využívány různé technologie (GPS, radiové vysílače, přírodní stopy, geolokátory atd.). Vlastníci datasetů a jejich spolupracovníci určují přístupnost a podmínky k využívání dat. Při vlastnictví příslušných práv jsou datasety dostupné ke stažení v mnoha datových formátech.

2.2 Předzpracování a transformace dat

Ač se to na první pohled nemusí zdát, velmi důležitá fáze procesu získávání znalostí z databází je selekce a předzpracování hrubých vstupních dat. V určitých případech dokonce může být nejpracnější. Její opomenutí pak může vést k pomalým výpočtům v dalších fázích či dokonce k chybným výsledkům.

2.2.1 Selekce dat

Objem dat, s kterými budeme pracovat, může být obrovský a z důvodů nastíněných již v úvodu lze očekávat, že se v budoucnu bude ještě zvyšovat. Je třeba oddělit data nesoucí užitečné informace od dat pro danou úlohu úplně zbytečných či pouze málo prospěšných. V souladu s našimi cíli provedeme analýzu dat a navrhneme automatické nástroje pro jejich redukci. V závislosti na kvalitě dat a navržených nástrojů pro nás mohou tato opatření znamenat různě velká urychlení výpočtů v následujících fázích.

Pokud např. chceme sledovat vzory a trendy pouze v určité lokalitě, měření provedená mimo tuto oblast jsou pro nás bezcenná a můžeme je tedy z dalších výpočtů vynechat.

2.2.2 Čištění dat

Žádná technologie pro sběr dat není bezchybná a dataset tak může obsahovat zjevně nesmyslná data, která mohou zkreslovat výsledky. Je vhodné tato data z datasetu vyjmout.

Když např. sledujeme trajektorii auta vybaveného GPS přijímačem, jsou jednotlivé body označené navazujícími časovými razítky v určitých nepříliš velkých vzdálenostech od sebe. Pokud se pak mezi nimi objeví záznam s polohou, na kterou by se v daném časovém intervalu nebylo možné dostat, jedná se o očividnou chybu měření. Ta může být způsobena např. vjezdem do tunelu apod.

2.2.3 Náhrada chybějících dat

Při práci s dostupnými datasety brzy zjistíme, že nám některá užitečná data chybí. To může být způsobeno chybným měřením, či jeho úplnou absencí (např. subjekt vědomě přístroj vypnul nebo ho naopak zapomněl v daném čase zapnout). V takovém případě může být vhodné tato data doplnit, odhadnout hodnotu chybějících měření z dat ostatních.

Je třeba si ale uvědomit, že takto doplněná data jsou pouze odhad a jeho přesnost se může lišit v závislosti na dostupných okolních měřeních, délce odhadovaného úseku, či použitých algoritmech. Obecně simulovaná data jsou vždy méně přesná a tedy horší než data reálná. Větší použití aproximovaných dat pak může vést ke zkreslení výsledků a chybným hypotézám na nich postavených. Častým neduhem je cílené doplňování dat tak, aby se potvrdila námi preferovaná a předpokládaná hypotéza.

Je důležité správně určit, jak velký úsek chybějících dat je vhodné ještě aproximovat a které algoritmy k tomu použít.

Lineární interpolace

Jednou z metod aproximace chybějících dat je lineární interpolace. Pokud jsou známé dva body p_0, p_1 o souřadnicích (x_0, y_0) a (x_1, y_1), výsledkem lineární interpolace je přímka, která je spojuje. Konkrétní interpolovaný bod v určitém čase pak na této přímce nalezneme pomocí časových razítek bodů p_0, p_1, viz [46].

Lineární interpolace je užitečná zejména při aproximaci malých úseků dat, při synchronizaci časových razítek, či pokud z nějakého důvodu potřebujeme rychlejší frekvenci záznamů lineárně se pohybujícího objektu (např. rychle jedoucího auta).

Aproximovaná data mají různou přesnost, kterou může být užitečné brát v potaz. V [28] autoři navrhují ke každé interpolované hodnotě přikládat pravděpodobnost ukazující věrohodnost odhadu. S touto pravděpodobností pak mohou pracovat další algoritmy ve fázi dolování z dat.

2.3 Dolování z dat

Dolování z dat je fáze, kdy za pomocí specializovaných technik a algoritmů, které si v této sekci popíšeme, získáme z předem připravených dat v určitém formátu požadovanou znalost. Ta se liší podle typu úlohy. V této sekci popíšeme obvyklý formát polohových dat, nejčastěji řešené úlohy dolování v polohových datech a známé techniky k jejich řešení.

2.3.1 Popis polohových dat

Většina datasetů obsahující polohová data má velmi podobnou strukturu. Pozice objektů jsou opatřeny *časovými razítky*, podle kterých jsou řazené. Tato časoprostorová data (angl. STD – Spatio-Temporal Data) můžeme formálně definovat takto:

Definice 2.3.1. $\text{STD} \subseteq \text{DLOC} \times \text{DTSTAMP} \times \text{DA}_1 \times \ldots \times \text{DA}_\text{k}$,

kde DLOC značí pozici objektu, DTSTAMP časové razítko a $\text{DA}_1 \times \ldots \times \text{DA}_\text{k}$ jsou další volitelné atributy.

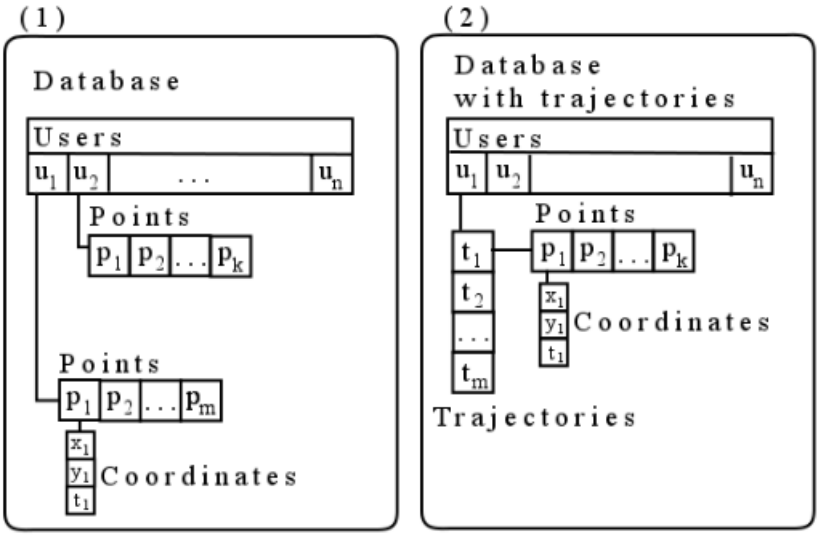
Pokud navíc přidáme atribut MOID, označující příslušnost těchto STD určitému objektu, získáme definici dat pohybujících se objektů (angl. MOD – Moving Object Data)

Definice 2.3.2. $\text{MOD} \subseteq \text{MOID} \times \text{DLOC} \times \text{DTSTAMP} \times \text{DA}_1 \times \ldots \times \text{DA}_\text{k}$,

V některých datasetech jsou navíc navazující data řazená do *trajektorií*, což jsou sekvence v nejjednodušším případě dvojic (loc, t) určujících polohu objektu v čase. Každá tato sekvence obsahuje identifikaci objektu, ke kterému přísluší a je uspořádaná podle času.

Definice 2.3.3. $tr_{oi} = \langle (loc_{oi1}, t_1), (loc_{oi2}, t_2), \ldots, (loc_{oin}, t_n) \rangle, \quad t_1 < t_2 < \ldots < t_n$

Obvyklou strukturu datasetu ilustruje obrázek 2.2.



Obrázek 2.2: Typická struktura datasetu obsahující polohová data: (1) bez trajektorií (2) s trajektoriemi.

2.3.2 Shlukování

Shlukování je proces, při kterém jsou objekty řazeny do předem neznámých tříd (shluků) na základě *podobnosti*. Základní podmínkou je, aby si objekty ve stejné třídě byly podobnější, než objekty v jiné třídě. Vzhledem k tomu, že jako podobnost lze definovat širokou škálu vlastností, je shluková analýza jedním ze základních konceptů využívaných při dolování v datech. V kontextu dolování v polohových datech lze shlukovou analýzu provádět např. na základě Euklidovské vzdálenosti, nicméně ne vždy je to výhodné. V [15] je euklidovská vzdálenost dvou bodů $X_1 = \{x_{11}, x_{12}, \ldots, x_{1n}\}$ a $X_1 = \{x_{21}, x_{22}, \ldots, x_{2n}\}$ definována následujícím vztahem:

$$dist(X1, X2) = \sqrt{\sum_{i=1}^{n}(x_{1i} - x_{2i})^2} \tag{2.1}$$

První publikace zmiňující shlukování dat vznikla před bezmála šedesáti lety [20] a od té doby vzniklo velké množství různých shlukovacích algoritmů využitelných pro různé účely.

Vzhledem k našemu zaměření na polohová data a plánovanému využití shluků jako vstupní data pro další etapy dolování se můžeme při výběru konkrétního shlukovacího algoritmu řídit určitými požadavky:

- *Škálovatelnost* – Některé shlukovací algoritmy pracují dobře pouze na malém objemu dat. Ten je ale u datasetů, s kterými budeme pracovat, zpravidla mnohem větší. Je tedy vhodné použít dobře škálovatelný shlukovací algoritmus.
- *Vytváření shluků různého tvaru* – Nejběžnější shlukovací metody směřují k vytváření kulovitých shluků podobné velikosti a hustoty. V některých případech ale objekty tvoří shluky jiných tvarů v závislosti na prostředí.

- *Vhodné vstupní parametry* – U většiny shlukovacích algoritmů je nutné nastavit určité vstupní parametry. Často ale vyžadují znalosti, které při jejich volbě nemáme – např. počet shluků.
- *Nezávislost na pořadí vstupních dat* – Některé shlukovací algoritmy mohou pro stejná data nalézt naprosto rozdílné shluky v závislosti na tom, jak je databáze uspořádána.

Následuje rozdělení shlukovacích metod do skupin a určení jejich výhod a nevýhod ve vztahu k našemu zaměření. Informace k tomuto rozdělení byly čerpány převážně z [15] a z [40].

Metody založené na rozdělování

Při dané databázi n objektů z nich shlukovací metoda založená na rozdělování vytvoří k tříd ($k \leq n$). Po skončení jejího průběhu pak každá třída tvoří jeden shluk. Data rozdělená do tříd musí společně splňovat následující podmínky:

1. Každá třída musí obsahovat alespoň jeden objekt.
2. Každý objekt musí patřit právě do jedné třídy (tato podmínka může být v některých variantách algoritmů zmírněna).

Při použití metody z této skupiny se vždy v prvním kroku určitým způsobem (náhodně nebo cíleně) vybere k objektů reprezentujících jednotlivé třídy. Do těchto tříd se na základě vzdálenostní funkce rozdělí zbylé objekty. Následně se v dalších iteracích v závislosti na konkrétním algoritmu vyberou nové objekty lépe reprezentující dané třídy a zbylé se opět přerozdělí.

Nejpopulárnějšími zástupci rozdělovacích shlukovacích algoritmů jsou algoritmy *k-means*, kde objekt reprezentující třídu je určen jako průměrná hodnota objektů do třídy náležících, a *k-medoids*, kde je jako reprezentující objekt určen vždy ten nejblíže centru shluku.

Tato skupina metod dobře pracuje s menšími a středně velkými množinami objektů. Nalézá pouze shluky kulovitého tvaru a počet těchto shluků musí být předem určen. Různé varianty se obvykle liší náročností výpočtů a vlivem odlehlých hodnot a šumu na výsledek.

Metody založené na hustotě

Tyto metody za shluky považují oblasti s vysokou hustotou objektů v prostoru, které jsou od sebe oddělené regiony s nízkou hustotou objektů. Ty jsou považovány za šum. Takový přístup umožňuje efektivní filtrování odlehlých hodnot a šumu a navíc možnost tvořit shluky libovolného tvaru.

Typickým zástupcem této skupiny je algoritmus DBSCAN [8]. Pro pochopení algoritmu je nutné zavést několik pojmů:

- Okolí objektu o poloměru ε se nazývá jeho ε-okolí.
- Pokud ε-okolí objektu obsahuje alespoň *MinPts* dalších objektů, nazývá se *jádro*.
- Při dané množině D, objekt p je *přímo dosažitelný na základě hustoty* z objektu q, pokud se p nachází v ε-okolí q a q je jádro.

- Při dané množině D, objekt p je *dosažitelný na základě hustoty* z objektu q, pokud existuje sekvence objektů $p_1, \ldots, p_n$, kde $p_1 = q$ a $p_n = p$ taková, že p_{i+1} je přímo dosažitelný na základě hustoty z p_i pro $1 \leq i \leq n, p_i \in D$.

- Při dané množině D, objekt p je *spojený na základě hustoty* s objektem q, pokud existuje objekt $o \in D$ takový, že p a q jsou dosažitelné na základě hustoty z o.

DBSCAN postupně prochází ε-okolí všech objektů v databázi. Pokud najde objekt p, v jehož ε-okolí se vyskytuje alespoň *MinPts* objektů, vytvoří nový shluk s p jako jádrem. Následně iterativně hledá objekty, které jsou z těchto jader přímo dosažitelné na základě hustoty, což může ústit ve spojování shluků. Proces končí, když žádný další bod nemůže být přidán do jakéhokoliv shluku.

Jak je patrné z popisu algoritmu, DBSCAN nevyžaduje předem daný počet shluků a kromě krajních bodů je nezávislý na řazení databáze. S použitím indexovací struktury je složitost algoritmu $O(n \log n)$. Různé nastavení vstupních parametrů *MinPts* a ε může mít zásadní vliv na výsledky. Pro data s různou hustotou je pak volba těchto parametrů problematická. Pro takové případy může být výhodnější použít příbuzný algoritmus OPTICS [2].

Daším příkladem shlukovacího algoritmu založeného na hustotě je DENCLUE [18].

Metody založené na mřížce

Tento přístup ke shlukování využívá víceúrovňovou mřížkovou datovou strukturu. Prostor, ve kterém se nacházejí objekty, rozděluje na konečný počet buněk, které jsou uspořádány do mřížky. Nad touto strukturou jsou prováděny všechny shlukovací operace.

Vzhledem k tomu, že výpočty nad touto strukturou nejsou závislé na počtu objektů, ale pouze na počtu buněk v jednotlivých úrovních mřížky, jsou velmi rychlé, což je hlavní výhoda tohoto přístupu.

Několik příkladů shlukovacích metod založených na mřížce:

- *STING* [38] – Buňky jsou hierarchicky rozděleny do několika úrovní, podle stupně rozlišení. Pro každou buňku jsou vypočteny statistické informace, které jsou v ní následně uložené a dále využívané při dotazování nad touto strukturou. Parametry buňky vyšší úrovně mohou být snadno odvozeny z parametrů buněk nižší úrovně.

  Výhodami této techniky jsou vzhledem ke mřížkové struktuře možnost paralelního zpracování a vysoká rychlost. Složitost formování hierarchické struktury je $O(n)$, kde n je počet objektů a složitost zpracování dotazu nad strukturou je $O(g)$, kde g je počet buněk mřížky na nejnižší úrovni, který je zpravidla mnohem nižší než n.

  Kvalita algoritmu je ve velké míře závislá na granularitě buněk nejnižší úrovně. Pokud je příliš jemná, doba výpočtů se dramaticky zvýší. Pokud je naopak příliš hrubá, kvalita shlukové analýzy je zásadně redukována.

- *WaveCluster* [35] – Využívá vlnkovou transformaci transformující původní datový prostor. Ta zdůrazňuje oblasti, kde dochází ke shlukování objektů v prostoru a zároveň potlačuje informace za hranicemi shluků.

  Tato metoda je velmi rychlá (složitost $O(n)$, kde n je počet objektů), nachází shluky libovolného tvaru, negativně ji neovlivňují odlehlé hodnoty, je nezávisla na seřazení databáze, nevyžaduje specifikaci vstupních parametrů a dobře pracuje s velkou množinou dat.

Metody založené na modelech

Cílem shlukovacích metod založených na modelech je při dané množině dat nalézt vhodný matematický model, který by jim odpovídal. Shodu se pak dále snaží optimalizovat. Tyto metody jsou často založené na předpokladu, že data jsou generována na základě složené pravděpodobnostní distribuční funkce.

Jako ukázku uvedeme algoritmus Expectation - Maximization (EM), který je souhrně popsán v [14].

V praxi může být každý shluk reprezentován parametrickou pravděpodobnostní distribuční funkcí. Celá datová množina tvoří směsici těchto funkcí. Na jejím základě můžeme vytvořit model hustoty o k distribučních funkcích, kde každá z nich reprezentuje jeden shluk. Populární algoritmus EM lze využít k nalezení parametrů těchto funkcí tak, aby co nejlépe odpovídaly daným datům.

EM algoritmus může být chápán jako rozšíření k-means. EM přiřazuje objekt ke shluku na základě hodnoty (váhy) reprezentující pravděpodobnost příslušnosti objektu k jednotlivým shlukům (krok expectation). Z těchto hodnot jsou vypočítány nové pravděpodobnosti (krok maximization).

Výhodou tohoto algoritmu je jeho snadná implementace. Rychle konverguje, ale ne vždy dosahuje globálního optima.

Dalšími využívanými metodami spadajícími do této skupiny jsou například *neuronové sítě* nebo *konceptuální shlukování*.

Hierarchické metody

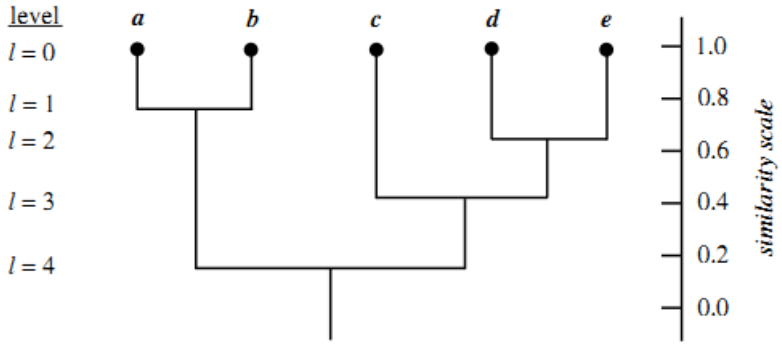
Hlavní myšlenkou hierarchických metod je vytvoření stromové struktury shluků. Existují dva přístupy k vytvoření takové struktury:

- *Aglomerativní přístup* (zdola-nahoru) – Každý objekt je umístěn do vlastní třídy. Následně jsou tyto atomické shluky spojovány do stále větších a větších shluků, dokud nevznikne jeden jediný shluk obsahující všechny objekty databáze anebo dokud nejsou splněny příslušné podmínky ukončující shlukovací proces.

- *Divizní přístup* (shora-dolů) – Zpočátku jsou všechny objekty umístěny do jediné třídy. Ta je rozdělována do stále menších a menších shluků, dokud každý objekt nepředstavuje vlastní třídu anebo dokud nejsou splněny ukončující podmínky (např. požadovaný počet shluků)

Proces hirarchického shlukování je nejčastěji reprezentován tzv. *dendrogramem*, jehož příklad můžeme vidět na obrázku 2.3. Jednotlivé úrovně indikují slučování shluků. Vertikální osa pak ukazuje měřítko podobnosti (vzdálenosti) tříd. Na tomto dendrogramu je také patrná základní nevýhoda čistě hierarchického přístupu, a to, že každé sloučení nebo rozdělení shluků je konečné. Proto se v praxi tento přístup často kombinuje s jinými (viz např. algoritmus BIRCH [41]).

Vzhledem k nevratnosti kroku určujícího sloučení/rozdělení shluků bývá tento krok výpočetně náročný a *přesné* hierarchické metody jsou tak špatně škálovatelné. Jako příklady čistě hierarchických shlukovacích algoritmů můžeme uvést algoritmy AGNES (aglomerativní přístup) a DIANA (divizní přístup).



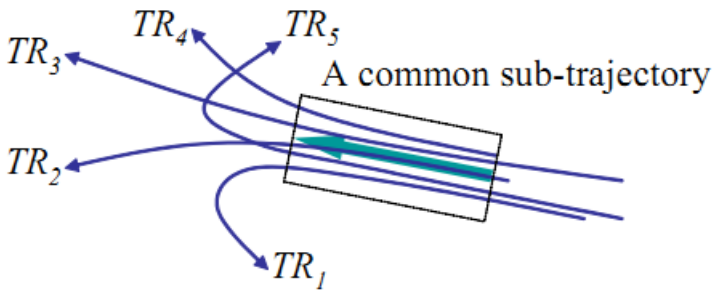
Obrázek 2.3: Dendrogram reprezentující proces shlukování objektů. Převzato z [15].

Shlukování trajektorií

S rozvíjejícím se výzkumem v této oblasti se objevily studie zabývající se shlukováním celých trajektorií.

Jedna z prvních prací na toto téma [11] ke shlukování trajektorii využívá metodu spadající do skupiny metod založených na modelech. Set trajektorii je popsán směsicí distribučních funkcí a členství ve shlucích je určováno pomocí EM algoritmu tak, jak bylo vysvětleno v příslušné části sekce 2.3.2.

V některých případech může ale být shlukování celých trajektorií na škodu. V [27] namítají, že trajektorie mohou být velmi dlouhé a společné chování objektů může být zachyceno pouze v jejich části (sub-trajektoriích), viz obrázek 2.4. Dále pak navrhují postup, jak tyto společné sub-trajektorie nalézt. Nejprve trajektorie rozdělí na menší části a ty pak shlukují.



Obrázek 2.4: Příklad společné sub-trajektorie. Převzato z [27].

Jedním ze základních problémů shlukování trajektorií je určení vzdálenosti mezi dvěma *kompletními* trajektoriemi. V [33] navrhují jako tuto vzdálenost použít průměr vzdáleností objektů vypočtených v jednotlivých periodicky naměřených časech. Dále představují algoritmus TF-OPTICS, který vyhledává časové intervaly nesoucí nejvíce informace a ty používá ke shlukování trajektorií, aby tak izoloval shluky vyšší kvality.

2.3.3 Vyhledávání vzorů

Cílem této úlohy je nalezení potenciálně užitečných vzorů chování v datech pohybujících se objektů.

Pohybové vzory

Jednou z oblastí zájmu je nalezení skupin objektů, pohybujících se společně v rámci určitého časového intervalu. V průběhu času bylo definováno mnoho skupinových pohybových vzorů. V [25] byly zavedeny jednoduché vzory popisující, zda-li skupina objektů konverguje k určitému cíli – vzor *konvergence* (angl. convergence), či se v něm dokonce střetne – vzor *setkání* (angl. encounter) nebo se v konkrétním čase pohybuje stejným či opačným směrem. Ty se vyvinuly ve více sofistikovaná *stáda* (angl. flock), *konvoje* (angl. convoy) a *roje* (angl. swarm).

Definice 2.3.4. Konvergence ($m > 1$, $r > 0$): Minimálně m objektů, které pokud budou udržovat současný směr pohybu, projdou určitým kruhovým regionem o poloměru r.

Definice 2.3.5. Setkání ($m > 1$, $r > 0$): Minimálně m objektů, které pokud budou udržovat současný směr a rychlost pohybu, se ocitnou *současně* uvnitř určitého kruhového regionu o poloměru r.

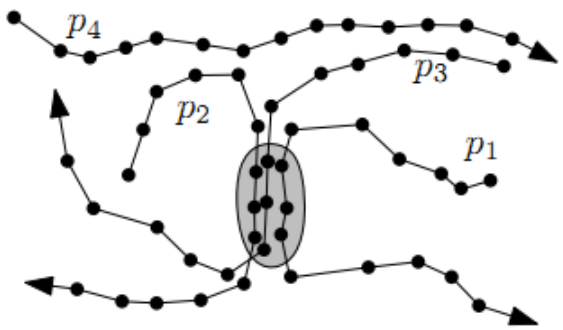
Stádo bylo poprvé představeno v [24] a dále studováno v [13, 12, 1], zejména ve směru jeho efektivního vyhledávání v datech. Definováno je takto:

Definice 2.3.6. Stádo ($m > 1$, $k > 0$, $r > 0$): Minimálně m objektů, které se v každém bodě v časovém intervalu o délce alespoň k nacházejí uvnitř kruhu s poloměrem r a pohybují se stejným směrem.

Jinými slovy, stádo je skupina objektů pohybující se společným směrem uvnitř *fixně* daného kruhu po *celou* dobu určitého časového intervalu. Na obrázku 2.5 je příklad stáda.

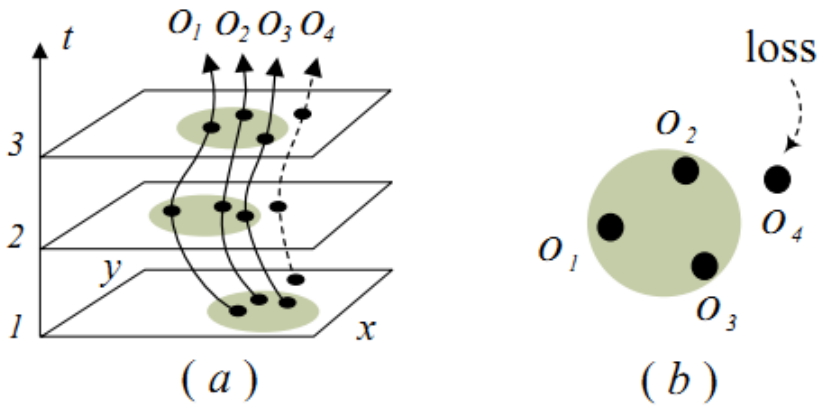
Definice stáda může být rozšířena na vzor *vedení* (leadership):

Definice 2.3.7. Vedení ($m > 1$, $k > 0$, $r > 0$, $s > 0$): Minimálně m objektů, které se v každém bodě v časovém intervalu o délce alespoň k nacházejí uvnitř kruhu s poloměrem r, pohybují se stejným směrem a alespoň jeden z těchto objektů se tímto směrem již pohyboval po dobu alespoň s časových kroků.



Obrázek 2.5: Stádo (vyznačeno šedě). Převzato z [12].

Jedna z hlavních nevýhod stáda je jeho omezení v podobě fixně daného kruhu uvnitř kterého musí všichni členové být po celý časový interval. Na obrázku 2.6 a) je patrné, že objekty cestují společně v přirozeně formované skupině, nicméně jak ukazuje 2.6 b), objekt o_4 není uvnitř stanoveného kruhu a není tak označen jako člen stáda. Určení velikosti poloměru kruhu při volbě vstupních parametrů algoritmu pro vyhledávání stád může být velmi problematické, protože skupiny objektů mohou být různě velké. V určitých případech je navíc kvůli rozložení dat kruhový tvar nevhodný.



Obrázek 2.6: Omezení pohybového vzoru stádo. Převzato z [22].

Konvoj, který je představen v [22, 21] a dále rozvíjen v [3] se s těmito omezeními úspěšně vypořádává. Můžeme ho definovat takto:

Definice 2.3.8. Konvoj ($m > 1$, $k > 0$): Minimálně m objektů, které se v každém bodě v časovém intervalu o délce alespoň k nacházejí ve své *blízkosti* a pohybují se stejným směrem.

Blízkost objektů je určena pomocí shlukování na základě hustoty v každém časovém kroku. Autoři ve svých studiích použili algoritmus DBSCAN, popsaný v sekci 2.3.2.

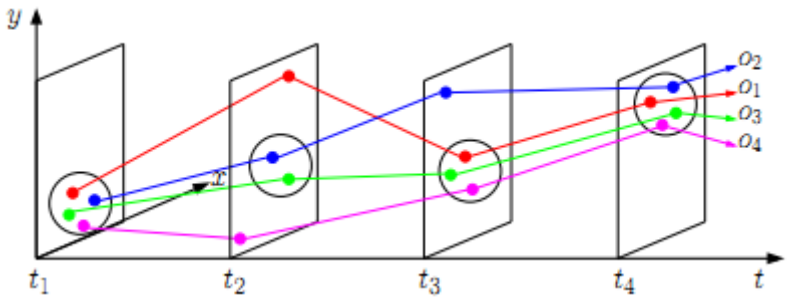
Roj [28] jde s volností definice pohybového vzoru ještě dál. Na obrázku 2.7 vidíme skupinu objektů a přestože někteří jedinci přechodně opouští shluk, intuitivně z dlouhodobého hlediska cestují společně. Všechny dříve navržené vzory mají jednu společnou nevýhodu. Objekty musí být ve stejném shluku po *jakékoliv* dva navazující časové kroky patřící do daného časového intervalu. Roj tuto podmínku nutné návaznosti pomíjí.

Definice 2.3.9. Roj (O, T): Jestliže $O_{DB} = \{o_1, o_2, \ldots, o_n\}$ je množina všech objektů, $T_{DB} = \{t_1, t_2, \ldots, t_n\}$ je množina všech časových okamžiků a $C_{DB} = \{C_{t_1}, C_{t_2}, \ldots, C_{t_n}\}$ je kolekce množin všech shluků v jednotlivých časových krocích $\{t_1, t_2, \ldots, t_n\}$, roj ($O$, T) představuje množinu $O \subseteq O_{DB}$ objektů, které jsou ve stejném shluku v každém časovém kroku z $T \subseteq T_{DB}$, kde $m > 0$, $n > 0$.

Periodické vzory

V mnoha případech se v periodicky opakujících časových intervalech (dny, týdny atd.) objekty pohybují po přibližně stejných trajektoriích. Z historických záznamů o jejich poloze lze tak odhadovat jejich zvyky a rutiny.

V práci [6], která se tímto tématem zabývá, definují periodický vzor takto:



Obrázek 2.7: Ztráta potenciálně užitečných pohybujících se shluků při použití vzoru stádo či konvoj. Převzato z [28].

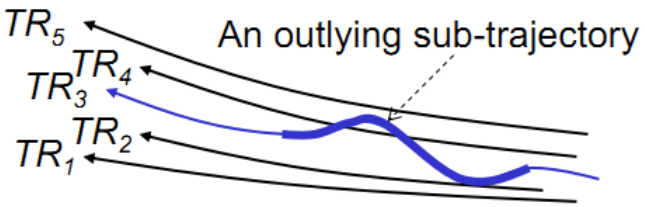
Definice 2.3.10. Periodický vzor P: Jestliže je daná sekvence $S = \{l_1, l_2, \ldots, l_n\}$ o délce n reprezentující velmi dlouhou historii pohybu objektu a sekvence $T = \{t_1, t_2, \ldots, t_m\}$ o délce $m << n$, kterou budeme nazývat *perioda* a která je daná, pak periodický vzor P definujeme jako sekvenci $\{r_1, r_2, \ldots, r_m\}$ o stejné délce jako T, která je součástí S více než min_{sup} krát.

2.3.4 Detekce odlehlé trajektorie

Opakem nalézání společných zvyků a pohybů objektů je detekce odlehlé trajektorie. Vyhledávání neobvyklého chování může být velmi užitečné zejména při odhalování kriminální činnosti (*deskriptivní úloha*) či k jejímu předcházení (*prediktivní úloha*). Dále může být zajímavé např. při objevování vzácných přírodních jevů.

Přístupy k řešení těchto úloh lze rozdělit na učení *bez učitele* a učení *s učitelem*.

Dále je třeba brát v potaz, zdali chceme detekovat celé odlehlé trajektorie nebo nás zajímají i pouze jejich části. Jak totiž podotýká [26], při detekci kompletní odlehlé trajektorie nám může unikat mnoho zajímavého neobvyklého chování (viz obrázek 2.8). Podobně jako při shlukování sub-trajektorií se neprve trajektorie rozdělí, a až pak probíhá detekce.



Obrázek 2.8: Příklad odlehlé sub-trajektorie. Převzato z [26].

Jedním z prvních přístupů, který spadá do přístupu učení bez učitele, bylo využití metody dobře známé ze shlukové analýzy – detekce na základě vzdálenosti [23]. Definice odlehlé trajektorie je pak následující:

Definice 2.3.11. Objekt $O \subseteq T$ považujeme za *odlehlý na základě vzdálenosti* (p, D), jestliže alespoň část p objektů z databáze T leží ve vzdálenosti větší než D od objektu O.

Jako příklad přístupu učení s učitelem můžeme uvést neuronové sítě, které autoři [34] využili k detekci podezřelého chování entit ve videích.

2.4 Vyhodnocení a využití získaných znalostí

Způsob vyhodnocení dolovací fáze závisí na našem cíli. Pokud testujeme vlastní přístup, či algoritmus, je vhodné k testování zvolit data, s kterými již někdo experimentoval, a výsledky porovnat. Tento krok je vhodný i pokud aplikujeme již známý přístup či implementujeme již publikovaný algoritmus, abychom se ujistili, že jsme postupovali správně.

Na kvalitu výsledků může mít vliv jakákoliv z fází KDD procesu. Použitý dataset, správné předzpracování dat v něm obsažených, výběr shlukovacího algoritmu, použité vzory a metody k jejich vyhledávání atd. Proto může být vhodné tento proces iterovat, některé volby obměnit a sledovat vliv těchto změn.

Výsledky dolování v pohybových datech jsou obvykle dále využívány v klasifikačních a predikčních systémech, případně ve studiích zabývajících se zvyky a trendy pohybujících se objektů. Jedna z oblastí využití znalostí z KDD procesu je rozpoznávání aktivit a cílů pohybujících se objektů, která bude popsána v další kapitole.

Kapitola 3

Rozpoznávání aktivit a cílů pohybujících se objektů

Vzhledem ke zvyšující se dostupnosti polohových dat se stále větší pozornost upírá k jejich možnému využití. Rozpoznávání aktivit a cílů objektů je jedno z nich. Rozpoznané aktivity mohou být základem pro real-time informační systémy, doporučovací systémy závislé na poloze, asistenční systémy pro fyzicky či mentalně postižené, systémy pro sdílení aktivit atd.

Tato kapitola je shrnutím dosavadního vývoje této oblasti. Nejprve se budeme zabývat úvodem do problematiky. Definujeme co je vstupem, cílem a které techniky se využívají k překlenutí mezery mezi těmito dvěma stavy. Popíšeme, s jakými daty se pracuje, jaké jsou možnosti a problémy při jejich získávání. Rozebereme hlavní překážky vyvstávající při rozpoznávání aktivit.

Samotný přehled a rozbor studií a dosavadně vyvinutých systémů následuje rozdělený do dvou kategorií podle toho, zda rozpoznáváme aktivity a cíle z dat jednotlivce, či z dat skupiny objektů.

3.1 Úvod do problematiky

Cílem této úlohy je extrakce *aktivit* a *cílů* (*vysokoúrovňová data*) jednoho nebo více tzv. *agentů* ze senzorových *měření* (*nízkoúrovňová data*) pomocí technik *dolování v datech* a *strojového učení*.

V systémů pro rozpoznávání aktivit a cílů tak musí být definovány rozpoznávané aktivity, požadovaná přesnost jejich rozpoznání a použité senzory pro sběr nízkoúrovňových dat.

Tyto systémy mohou pracovat s trajektoriemi pouze jednotlivce či se mohou zabývat rozpoznáváním aktivit a cílů více objektů a zjišťovat vztahy a odlišnosti uvnitř céle skupiny.

Samotné metody rozpoznávání pak lze rozlišit podle metody učení na učení s učitelem a učení bez učitele.

3.1.1 Použitá data a jejich měření

Typ senzoru lze vybírat na základě následujících kritérií:

- *Počet* – Můžeme volit např. mezi jedním velkým, efektivním senzorem nebo větším počtem menších.

- *Typ dat* – Senzory zachycující video, zvuk, pohyb, polohu atd.
- *Umístění* – Senzor může být umístěn na objektu. Pak je třeba brát ohled na pohodlí objektu, na spotřebu energie a robustnost senzoru. Další možností je umístění v prostředí, v kterém se objekt pohybuje. V tomto případě musíme brát v potaz pokrytí senzoru a jeho rezistenci vůči vnějším vlivům.

Rozpoznání aktivit určitého objektu je také možné nepřímo z pozorování aktivit příbuzných objektů. Typy a umístění senzorů může být výhodné kombinovat za účelem zpřesnění výsledků.

3.1.2 Problémy a překážky při rozpoznávání aktivit

Vzhledem k tomu, že masivní nárůst polohových dat pozorujeme až v posledních letech, je většina navržených systémů z této oblasti otestována pouze v omezených, často laboratorních podmínkách. Přestože očekávání do budoucna jsou velice optimistická, v současnosti je problematické zajistit ideální podmínky pro testování systémů rozpoznávajících aktivity. Ty tak mají pouze omezenou funkčnost. Např. velmi omezený set rozpoznávaných aktivit, pracují pouze v určitém prostředí atd.

Jednou z prvních překážek, na kterou tyto systémy naráží, je sběr dat. GNSS systémy mají problém s pozicováním uvnitř nebo v okolí budov, bezdrátové technologie nemusí mít dostatečné pokrytí, senzory umístěné na objektech mohou být pro tento objekt nepříjemné atd. Nositelné senzory jsou v současnosti také často prototypy, které pracují pouze v krátkých časových intervalech, snadno se ničí atd.

Dizertační práce [30] se věnuje překažkám souvisejícím s učením systému a kombinováním dolovaných informací za účelem spolehlivého rozpoznání aktivit. Všímá si propasti mezi nízkoúrovňovými senzorovými daty a rozpoznávanou aktivitou. K překročení této propasti navrhuje kombinovat znalosti získané pomocí technik dolování z dat. Např. ke spolehlivému odhadu cíle pohybujícího se objektu využívá pohybové vzory a odhady cesty společně s rozpoznáváním transportního módu a spojením trajektorií s cestovními mapami.

Způsoby provádění jedné, stejné aktivity u živých objektů jsou pro každého jedince specifické, závislé na jeho schopnostech a preferencích. Navíc se mohou neočekávaně měnit vlivem vnějších faktorů. Systém k jejich rozpoznávání tak musí pracovat s velmi vágní definicí jednotlivých aktivit. V některých případech může být vhodné se zaměřit na rozpoznávání kombinace aktivit a jejich návaznost namísto striktně separovaného přístupu.

3.2 Rozpoznávání aktivit a cílů z dat jednotlivce

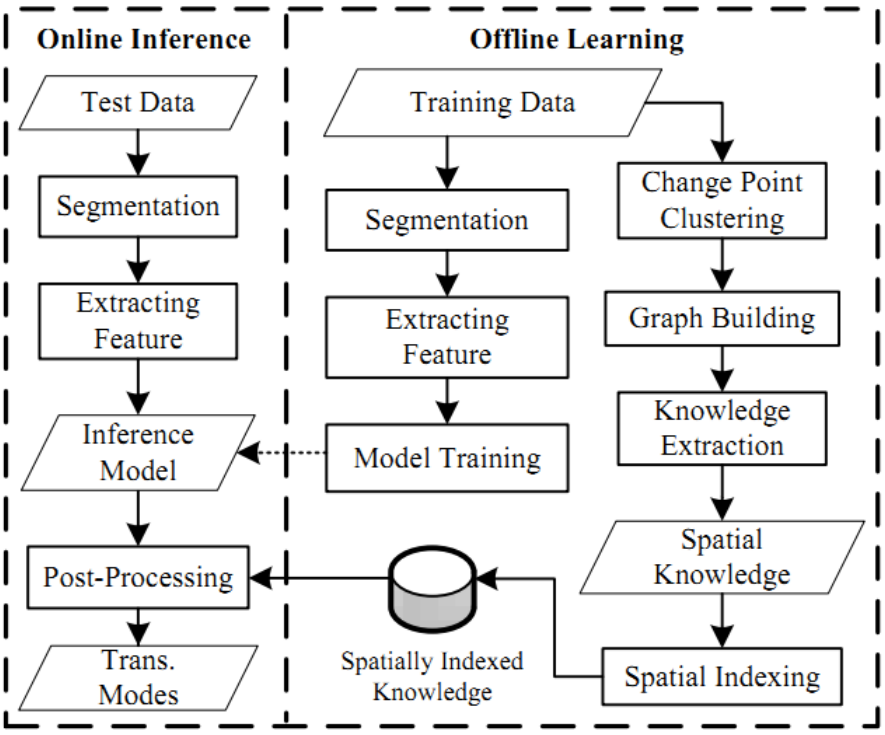
Metody rozpoznávání aktivit z dat jednotlivce si rozdělíme do dvou skupin podle metody učení.

3.2.1 Učení s učitelem

Metoda učení s učitelem v první fázi jako vstup pro trénovaný klasifikační či regresní model používá trajektorie pohybujícího se objektu společně s korespondujícími štítky (angl. label) popisujícími aktivity vykonávané v příslušném čase. Tento vstup nazýváme *trénovací* sadou dat. V druhé fázi pak tento model dokáže automaticky vytvářet štítky s aktivitami pro *testovací* sadu dat (pouze samotné trajektorie).

Odvození transportního módu

Ve vztahu k polohovým datům je jedna z intuitivních aktivit k rozpoznání transportní mód, tedy prostředek použitý k přesunu. Autoři projektu GeoLife představili v [43] metodu spadající do skupiny učení s učitelem, která se odvozování transportního módu věnuje. Její architekturu lze vidět na obrázku 3.1.



Obrázek 3.1: Architektura systému pro rozpoznávání transportního módu. Převzato z [43].

V souladu s typickým schématem metody učení s učitelem můžeme vidět, že architektura tohoto přístupu se skládá ze dvou částí: offline učení a online odvození módu.

V učící části je trénovací dataset trajektorií rozdělen do segmentů na základě bodů, v kterých objekt změnil mód. Z těchto segmentů jsou následně extrahovány vektory příznaků. Ty jsou využity k trénování klasifikačního modelu. V odlišném procesu jsou body, ve kterých objekt změnil transportní mód, shlukovány na základě hustoty a zabudovány do grafu s pomocí uživatelských trajektorií. Graf je využíván k extrakci znalostí založených na poloze (např. pravděpodobnostní rozložení transportních módů na hranách grafu), které zvyšují přesnost výsledku ve fázi post-zpracování.

V online nasazení je příchozí trajektorie opět rozdělena do segmentů, z nichž jsou vy-extrahovány vektory příznaků. Trénovaný klasifikační model z nich následně odvodí transportní mód pro každý segment. Tento odhad je zpřesněn s přihlédnutím k pravděpodobnosti náležitosti módu do daného segmentu.

Samotný odhad módu je založen na rozhodovacím stromu, který podle studie obsažené v [44] přináší přesnější výsledky odhadu transportního módu než modely CRF, SVM (Support Vector Machine) a Baysovské sítě. Dynamické Bayesovské sítě jsou k tomuto účelu využívány např v [30].

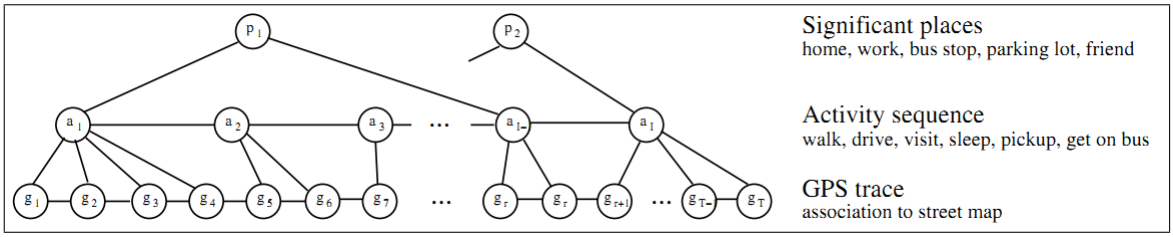
Rozpoznávání aktivit a důležitých míst

Liao *aj.* [31] ukazují možnost využití hierarchického modelu k rozpoznávání aktivit a důležitých míst na základě polohy. Na obrázku 3.2 tak můžeme vidět tři úrovně modelu:

- *GPS trajektorie* – Definice viz sekce 2.3.1. Pro zvýšení úspěšnosti v rozpoznávací fázi jsou trajektorie segmentovány a zarovnány do úseků ulic.
- *Aktivity* – Pro jednotlivé úseky ulic jsou rozpoznány aktivity jako je „spánek“, „chůze“, „řízení“ atd. Ty jsou dále použity k odhadu důležitých míst.
- *Důležitá místa* – Místa, která jsou úzce spjatá s aktivitami objektů, jako je domov, škola, autobusová zastávka, parkoviště atd.

Metoda popsaná v [31] se skládá ze dvou krok;. První z nich uspořádává GPS data do segmentů a ty dále zarovnává do úseků ulic. K tomu využívá diskriminativní model CRF (Conditional Random Fields).

V druhém kroku se z jednotlivých úseků ulic extrahují příznaky, jako je průměrná rychlost, denní doba atd. Ty se využívají k odvození aktivit a důležitých míst s pomocí nového CRF modelu. Hlavní překážkou k trénování tohoto CRF modelů je absence štítků označujících důležitá místa v trénovací sadě dat. K jejich nalezení autoři doporučují využít EM algoritmus.



Obrázek 3.2: Hierarchický model pro rozpoznávání aktivit na základě polohy. Převzato z [31].

3.2.2 Učení bez učitele

Hlavní výhodou učení bez učitele je absence jakékoliv trénovací fáze. Tato metoda nevyžaduje žádné štítky, pracuje pouze s trajektoriemi. Snaží se v datech nalézt užitečné vzory aktivit. Výsledkem tak nebývají přímo štítky s aktivitami, ale spíše znalosti, které mohou aktivity implikovat.

Využití shlukové analýzy při rozpoznávání aktivit

Typickým zástupcem metod učení bez učitele je shluková analýza, kterou jsme popsali v sekci 2.3.2. Ve většině systémů pro rozpoznávání aktivit se pracuje s vektorem příznaků. Huynh *aj.* [19] vytvořili hodnotící systém, který za pomoci shlukové analýzy vybírá nejvhodnější znaky pro potřeby rozpoznávání aktivit.

Extrakce denních rutin

Součástí projektu MIT RealityMining[1] je volně dostupný dataset obsahující telefonní záznamy sta zaměstnanců a studentů fakulty za devět měsíců. Záznamy každého jedince obsahují telefonní volání, krátké zprávy, status telefonu, blízkost bluetooth zařízení a identifikátor telefonní věže. Identifikátor telefonní věže určuje přibližnou polohu objektu a umožňuje odvodit některé základní stavy objektu: „v práci“, „doma“, „bez signálu“ atd.

S tímto datasetem pracuje projekt Eigenbehavior [7]. Ten pro každého uživatele vytváří datovou sadu H-dimensionálních binárních vektorů $\Gamma = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_D\}$, kde D je počet dní. Každý bit vektoru Γ_i indikuje stav objektu v danou hodinu. Průměrné (rutinní) chování je pak odvozeno pomocí PCA (Principle Component Analysis).

Další metodou extrakce denních rutin využívající výše uvedený dataset je LDA (Latent Dirichlet Allocation), která byla původně navržena k automatizovanému odvození tématu z bloku textu [5].

Dny jsou podle přístupu navrženého v [9] rozděleny do třicetiminutových časových oken a ke každému oknu je přidělen štítek popisující stav uživatele takový, který se v daném časovém intervalu objevoval nejvíce. Den je pak brán jako směsice navazujících stavů, které jsou zakódovány do tzv. *slov* společně s identifikátorem časového okna. LDA algoritmus v těchto slovech nalézá denní rutiny. Čím častěji se určitá kombinace stavů objevuje, tím pravděpodobněji bude zakódovaná do výsledné rutiny.

Dolování frekventovaných vzorů

Dolování sekvenčních vzorů aktivit se prolíná s oblastí dolování pohybových vzorů, kterou jsme rozebírali v sekci 2.3.3. Periodické vzory pak tvoří průnik těchto dvou oblastí.

Jedna z posledních studií na toto téma [29] se zabývá nalezením periodických vzorů aktivit v trajektoriích zvířat. Jejich metoda se skládá ze tří kroků:

1. *Nalezení pozorovaných míst* – Mapa je rozdělena do mřížky. Pro každou buňku v mřížce je vypočítána hustota GPS dat v ní obsažených. Následně je pozorované místo složeno z buněk o hustotě vyšší, než je zvolený práh, viz obrázek 3.3.

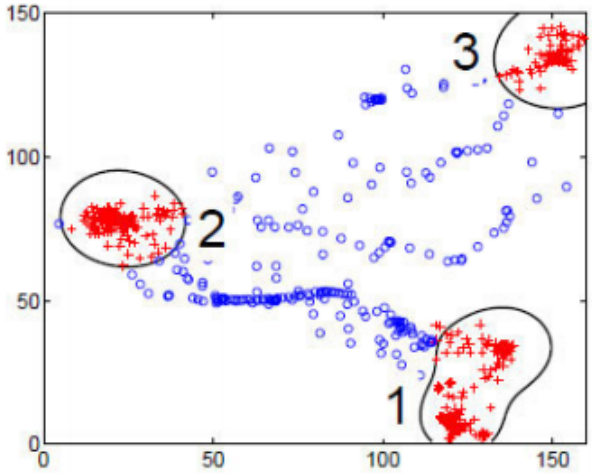
2. *Detekce period na základě binárních pohybových sekvencí* – Pro každé pozorované místo je pohyb objektu transformován do binární sekvence, kde 1 značí, že je objekt na pozorovaném místě a 0, že není. Periody mohou být následně nalezeny za pomoci Fourierovy transformace a autokorelace.

3. *Dolování periodických vzorů* – Podle periody (např. den) je pohyb objektu rozdělen do segmentů. Některé segmenty v průběhu času tvoří periodické chování (denní chování v létě, v zimě aj.). To lze nalézt pomocí aglomerativního hierarchického shlukování.

3.3 Rozpoznávání aktivit a cílů z dat skupiny objektů

Rozpoznávání aktivit a cílů z dat skupiny objektů přináší celou řadu nových překážek. Při trénování klasifikačního modelu pomocí dat pocházejících od různých uživatelů musíme přihlédnout ke způsobu, jakým objekty danou aktivitu provádějí. Pokud neexistují velké rozdíly v preferencích či schopnostech objektů, lze použít některý z modelů popsaných

[1] http://realitycommons.media.mit.edu/



Obrázek 3.3: Pozorovaná místa. Převzato z [46].

v předchozí sekci. V ostatních případech je třeba brát v potaz vztahy mezi objekty a aktivitami jimi prováděnými.

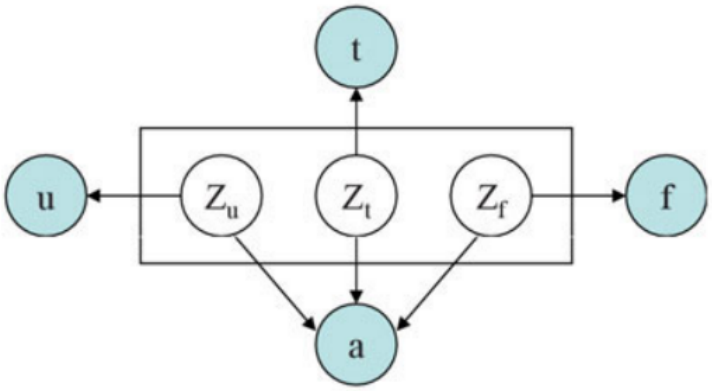
3.3.1 Využití dat více uživatelů k přesnějšímu rozpoznání aktivity jednotlivce

V rámci projektu Darwin Phones [32] si všímají faktu, že přestože se dva mobilní telefony nachází ve stejném prostředí, jejich senzorová informace může být odlišná. Např. telefon uvnitř batohu označí prostředí, v kterém se vyskytuje jako tiché, zatímco senzory používaného telefonu v rukou uživatele budou indikovat hlasitější prostředí. Při online rozpoznávání aktivit jsou proto k upřesnění odhadu aktivity využívány i informace z okolních mobilních telefonů.

Autoři [42] navrhují personalizovaný model využívající data skupin objektů. Namísto použití všech dat k trénování jednoho modelu zavádějí tzv. *aspektové* uživatelské proměnné určené k zachycení vztahu uživatele ke skupině. Při rozpoznávání aktivit cílového uživatele lze tak využít data podobných uživatelů ze stejné skupiny a zvýšit tak přesnost odhadu. S tímto přístupem je také zásadně redukována potřeba oštítkovaných dat v trénovací fázi.

V prostředí WiFi jsou sbírána data ve formátu: $\{\langle a_i, u_i, f_i, t_i\rangle | i \in \{1, \ldots, L\}\}$, kde a_i je aktivita, u_i je uživatel a f_i je informace o AP (Access Point) v čase t_i. Jeden záznam tak indikuje, že uživatel u_i vykonává aktivitu a_i v čase t_i a jeho bezdrátové zařízení detekuje AP f_i.

Obrázek 3.4 představuje grafický model. Kromě již zmíněných stínových proměnných vidíme i aspektové uživatelské proměnné $Z_u \in \{z_u^1, z_u^2, \ldots, z_u^{D_u}\}$ indikující D_u uživatelských shluků. Uživatelé ze stejných shluků se mohou společně podílet na trénování klasifikátoru za pomoci svých t a f informací. Přestože má pak např. jeden uživatel limitované množství dat pro potřeby tréninku klasifikátoru, stále může využívat dat ostatních uživatelů ve stejném shluku. Jsou zavedeny i další aspektové proměnné $Z_t \in \{z_t^1, z_t^2, \ldots, z_t^{D_t}\}$, $Z_f \in \{z_f^1, z_f^2, \ldots, z_f^{D_f}\}$, které jsou využívané k zachycení vztahů mezi aktivitami.



Obrázek 3.4: Aspektový model. Převzato z [42].

3.3.2 Rozpoznávání souběžných aktivit

V [37] představují systém určený k rozpoznávání souběžných aktivit. Data každého uživatele se skládají ze sekvence pozorování, která jsou opatřena štítkem označujícím prováděnou aktivitu. Aby mohli modelovat interakci a vzájemný vliv uživatelů v závislosti na čase, autoři [37] využívají rozšíření HMM v podobě CHMM (Coupled Hidden Markov Model).

Kapitola 4

Návrh vlastního systému pro rozpoznávání aktivit

Tato kapitola se věnuje návrhu vlastního systému pro rozpoznávání aktivit kombinujícího některé metody popsané v předchozích dvou kapitolách. Nejprve si ve stručnosti popíšeme architekturu systému a následně se budeme podrobněji zabývat jeho jednotlivými součástmi.

4.1 Architektura systému

Systém si klade dva hlavní cíle:

1. Nalézt rutinní lidské chování v určitých časových intervalech (periodách) pouze z dat GPS trajektorií. Reprezentovat je pomocí periodických vzorů.

2. Určit, zda-li nové záznamy uživatele spadají do některé z dříve nalezených rutin. Jinými slovy klasifikovat nové GPS trajektorie na základě dříve vytvořených periodických vzorů.

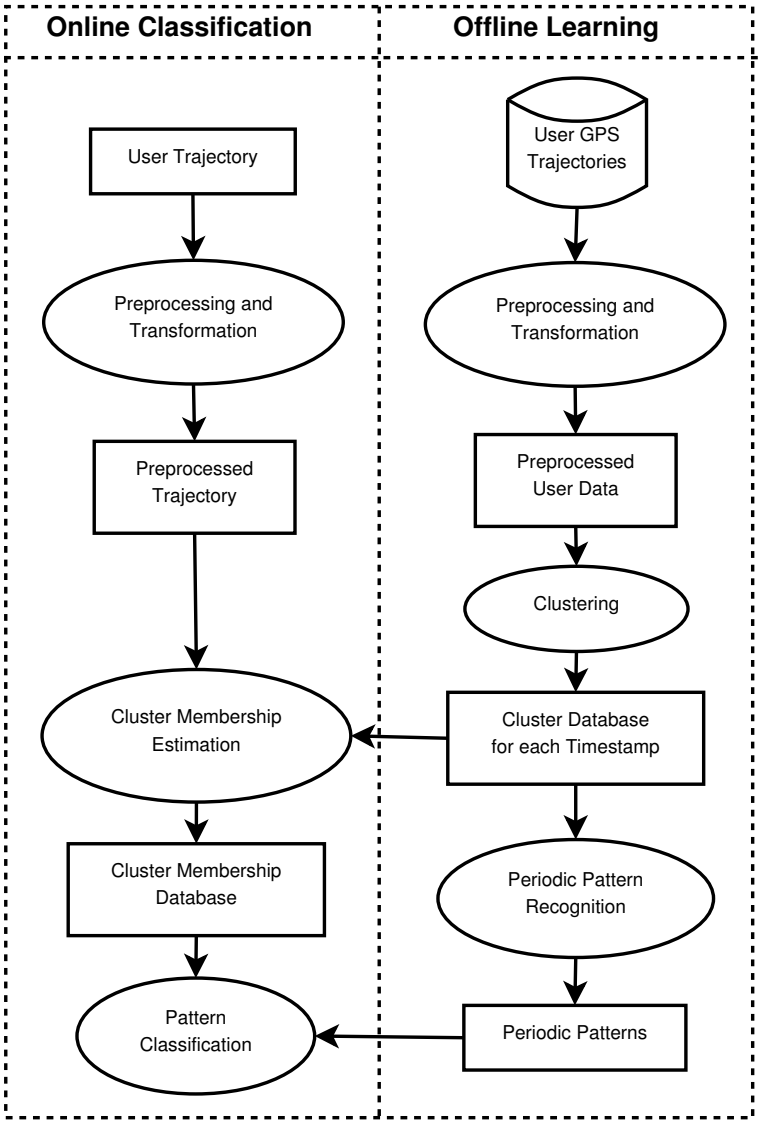
Znalosti o rutině jednotlivce mohou být užitečné ke zlepšení přesnosti dalších systémů pro rozpoznávání aktivit, k detekci nestandardního chování, vedení deníku aj.

Na obrázku 4.1 je vykresleno schéma architektury systému, rozdělené do dvou částí dle vytyčených cílů a způsobu zpracování.

Vpravo vidíme offline zpracování kompletní historie uživatele. Na ní je aplikován KDD přístup, popsaný v kapitole 2. Datovou sadu po předzpracování na základě zvolené periody transformujeme. Nejprve body rozdělíme na úseky odpovídající dané periodě a následně tyto úseky překryjeme. Vznikne tak sada menších datasetů, která má délku jedné periody složené z časových okamžiků, kde v každém okamžiku jsou obsažené body z různých úseků. Body rozdělené podle časových okamžiků následně shlukujeme. Pro každý časový okamžik tak vznikne množina shluků. Tato databáze množin shluků je vstupem pro algoritmus vyhledávající samotné periodické vzory. Zároveň je také využívána při online zpracování vstupních trajektorií. Výsledkem této offline větve systému je sada vzorů reprezentujících periodické chování při zvolené periodě.

Vlevo je pak ilustrována metoda přístupu k nové trajektorii uživatele, kterou lze za předpokladu předem vytvořené databáze shluků a vzorů provádět online. Podobně jako v předchozím případě nejprve provedeme předzpracování dat. Následně využijeme databázi

shluků a označíme ty, do kterých by na základě vzdálenosti mohly patřit body z námi zpracovávané trajektorie. Samotná klasifikace je pak založena na posobném principu jako algoritmus pro vyhledávání vzorů.



Obrázek 4.1: Architektura navrženého systému.

4.2 Výběr datové sady

Abychom mohli otestovat náš systém, musíme vybrat vhodnou datovou sadu. Nejprve si shrneme požadavky na testovací datovou sadu a následně v souladu s nimi provedeme výběr.

Požadavky na datovou sadu

- *Polohová data lidí* – Práce se zaměřuje na vyhledávání rutinního *lidského* chování. Vhodní lidští zástupci jsou na rozdíl od zvířat nebo přírodních živlů více organizovaní,

jejich den je strukturovanější, přesněji vnímají čas, kterému se navíc vědomě podřizují. Periody tak budou pravděpodobně ostřeji vymezeny a budou se méně překrývat. Lze navíc předpokládat, že lidské periody chování budou kratší a pravidelnější, protože jsou méně ovlivněné přírodními podmínkami.

- *Dlouhý časový interval* – Pro otestování našeho systému není nutné velké množství různých objektů, nicméně je vhodné vybrat datovou sadu posbíranou v rozmezí delšího časového intervalu, která umožní přesnější odhad. Ideálně pak data shromážděna v rámci několika let. Lze tak detekovat peridické chování ve specifických každoročně se opakujících událostech (svátky, festivaly atd.).
- *Vhodný format dat* – Data by měla být pravidelně strukturovaná tak, aby s nimi šlo snadno pracovat pomocí standardních technik práce se soubory.
- *Volně dostupná* – Vzhledem k nekomerčnosti našeho projektu lze vybírat pouze z volně dostupných, neplacených datových sad.

4.2.1 Datová sada GeoLife GPS Trajectories

Datovou sadou, která splňuje všechny výše uvedené požadavky je již dříve (viz sekce 2.1.1) zmiňovaný dataset GeoLife GPS trajectories, který zachycuje GPS trajektorie lidí v oblasti Pekingu v délce několika let a je tak ideální pro vyhledávání periodického chování. V budoucím vývoji může být výhodou, že část dat je opatřena štítky indikujícími transportní mód objektu, které lze využít např. k trénování klasifikátoru. Obrázek 4.2, převzatý z dokumentace k této datové sadě, ilustruje rozložení dat (tzv. tepelnou mapu).

Výběr konkrétního objektu

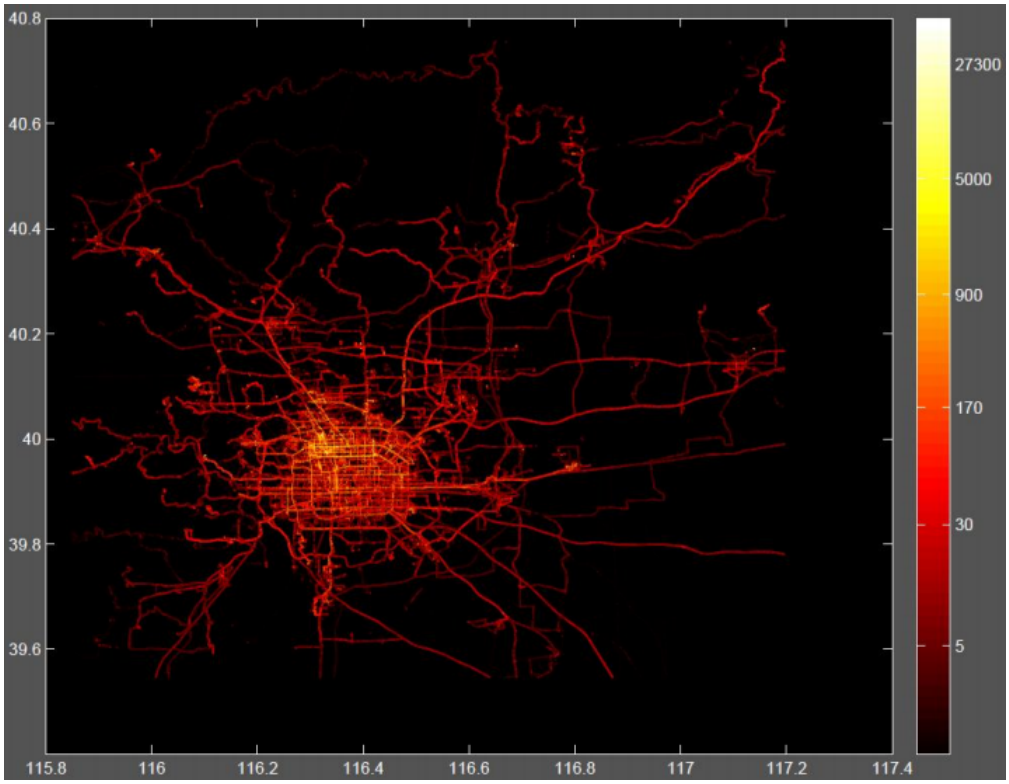
Ke zpracování je nutné vybrat objekty vykazující periodické chování. Např. nelze předpokládat úspěch obecného systému pro rozpoznávání periodického chování z polohových dat u člověka cestujícího kolem světa. Ideálním kandidátem je objekt s pevným denním časovým harmonogramem (pevná pracovní doba apod.). Vzhledem k tomu, že tyto preemptivní znalosti nejsou součástí vybrané datové sady, budeme kandidáty vybírat pouze na základě množství polohových dat jednotlivých objektů a časového intervalu, v kterém byly pořízeny.

4.3 Předzpracování datové sady

Datovou sadu je vhodné předzpracovat tak, aby co nejlépe vyhovovala potřebám systému. Velká část těchto operací byla popsána v sekci 2.2, nicméně každý dataset je do jisté míry jedinečný a přináši vlastní problémy a překážky.

4.3.1 Odstranění chybných dat

Jednotlivé polohy objektů jsou zaznamenány v pravidelných časových intervalech t_{vz}. V podobě bodů jsou pak za sebou řazeny do trajektorií (viz obrázek 2.2 vpravo). Pokud jsou dva body trajektorie v časových okamžicích t_i a t_{i+1} od sebe v čase vzdálené t_{vz} sekund, ale geograficky vzdálené více než práh l_{max} označující maximální možnou vzdálenost, kterou je obyčejný člověk schopný urazit v čase t_{vz}, jeden z těchto dvou bodů je chybný. Ze vzdálenosti bodů t_{i-1}, t_i a t_{i+1}, t_{i+2} zjistíme konkrétní chybný bod a vzhledem k tomu, že takový bod je pro naše potřeby bezcenný, z testovací datové sady ho odstraníme. V případě



Obrázek 4.2: Rozložení polohových dat v datové sadě GeoLife GPS trajectories. Obrázek byl převzat z dokumentace k datové sadě.

větší chyby (např. cesta podzemím apod.) aplikujeme podobný postup na širší časové okolí bodů.

4.3.2 Řešení chybějících dat

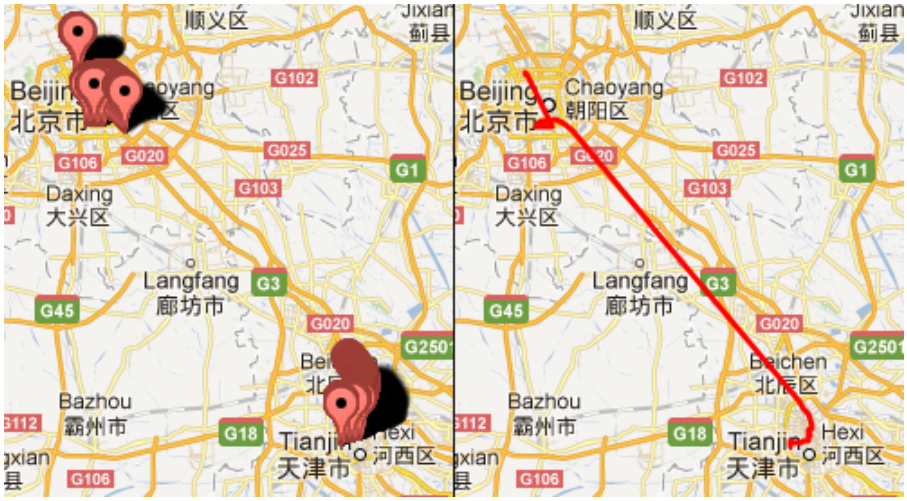
Pokud jsou dva body trajektorie v časových okamžicích t_i a t_{i+1} od sebe v čase vzdálené více než t_{vz}, v trajektorii chybějí data. Počet chybějících vzorků T_N lze spočítat z následujícího vztahu:

$$T_N = \frac{t_{i+1} - t_i}{t_{vz}} \tag{4.1}$$

Na obrázku 4.3 jsou reálná data z námi vybrané datové sady, konkrétně jedna trajektorie jednoho objektu z jednoho dne. Vlevo je trajektorie vykreslená po jednotlivých bodech, vpravo jsou body spojené v křivku, která simuluje lineární interpolaci.

Je patrné, že velká část bodů, které by spojovaly dvě viditelně prostorově vzdálené subtrajektorie, chybí. GPS zařízení bylo zřejmě podstatnou část dne vypnuté a my si můžeme jen domýšlet, kde se objekt pohyboval. Při takovém množství chybějících dat bychom lineární interpolací vnesli do dat velkou nepřesnost, která by mohla zapříčinit zkreslení výsledků. V této situaci je vhodnější trajektorii rozdělit na dvě menší.

Zavedeme tedy pravidlo, že pokud bude počet chybějících vzorků T_N větší než práh t_{max}, rozdělíme trajektorii na dvě menší. V opačném případě chybějící body doplníme pomocí lineární interpolace.



Obrázek 4.3: Trajektorie vykreslená po bodech (vlevo). Trajektorie vykreslená jako křivka (vpravo)

4.4 Transformace dat

Při práci s datovou sadou ji bude v některých krocích procesu nutné transformovat. Data bude třeba přizpůsobit shlukovacím a dolovacím algoritmům, změnit jejich reprezentaci apod.

4.4.1 Volba periody

Logickým krokem při detekci periodického chování je volba periody s kterou budeme pracovat. V [29] je představena metoda, která tento krok provádí automaticky, na základě dat. To může být užitečné zejména u zvířat a přírodních živlů, kde se periody častěji mění, prolínají a nejsou ostře ohraničené.

U lidských objektů si vystačíme s pevným nastavením periody v souladu s přirozenými přírodními cykly (den, měsíc, rok) a případně dále pak s rozdělením vzorků podle lidských zvyků (např. pracovní dny, víkendové dny).

Na základě námi zvolené periody následně rozdělíme data na stejně dlouhé úseky. Pro potřeby shlukové analýzy a dolovacích algoritmů všem bodům přiřadíme identifikátor příslušného úseku. Na data spadající do různých úseků o stejné periodě pak budeme z pohledu dalších kroků procesu nahlížet jako na data odlišných objektů. Úseky následně překryjeme. Vznikne tak databáze o délce jedné periody rozdělená do jednotlivých časových okamžiků. V každém se tak nalézají příslušná data z různých úseků.

Pokud $S = \{p_0, p_1, \ldots, p_{N-1}\}$ je sekvence bodů označující historii polohy uživatele, pak s volbou periody P je tato historie rozdělena do $m = \frac{N}{P}$ úseků, kde každý úsek U obsahuje množinu bodů $\{p_i, pi + P, \ldots, p_{i+(m-1)\cdot P}\}$ pro $0 \leq i \leq P$. Každému bodu je přidělen identifikátor $id \in 0, \ldots, m-1$.

Jinými slovy, na základě námi zvolené periody rozdělíme data na stejně dlouhé úseky. Pro potřeby shlukové analýzy a dolovacích algoritmů všem bodům přiřadíme identifikátor příslušného úseku. Na data spadající do různých úseků o stejné periodě pak budeme z pohledu dalších kroků procesu nahlížet jako na data odlišných objektů. Úseky následně

překryjeme. Vznikne tak databáze o délce jedné periody rozdělená do jednotlivých časových okamžiků. V každém se tak nalézají příslušná data z různých úseků.

4.4.2 Řešení časových posunů

V reálném světě je často periodické chování do jisté míry posunuté v čase. Objekt tak např. denodenně vykazuje stejný vzor chování, nicméně v některých dnech vstane dříve, v některých dnech zaspí. Může se dostat do zácpy a přijít do práce později apod. Vzhledem k tomu že posunuté mohou být i jen části periody, je jednoduchý posun všech bodů v čase neúčinný. Odchylky v čase představují velkou překážku při dolování periodických vzorů.

Zarovnání bodů v čase

Pro potřeby shlukové analýzy je nutné nejenom, aby byly mezi body stejné časové rozestupy, ale také, aby byly body v čase synchronizovány. Je tedy třeba pro každou periodu stanovit začátek a pevný časový krok t_{step}. Následně každý bod spadající do příslušné periody posuneme *v čase* k nejbližšímu referenčnímu bodu.

Pokud zvolíme $t_{step} > t_{vz}$, dostaneme v nově vytvořených časových okamžicích více polohových bodů. Toto násilné snížení počtu časových okamžiků při zachování stejného množství polohových bodů může posun periodického chování v čase částečně vyrovnat.

Kopírování bodů do okolí

V [6] je popsána metoda kopírování bodů, která problém časového posunu do jisté míry řeší. Tato metoda počítá s rozdělením všech bodů do menších datasetů v závislosti na svých časových okamžicích (viz sekce 4.4.1). Ty jsou předem synchronizovány a zarovnány tak, jak je popsáno v minulé podsekci. Proces shlukování, popsaný v následující sekci, pak bude postupně použit na každý z těchto datasetů. Aby bylo zajištěné, že budou brány v potaz i v čase posunuté body, experimentálně zvolíme konstantu τ, která bude značit časovou vzdálenost, v rámci které bude každý bod v každém časovém okamžiku rozkopírován. Tuto časovou oblast budeme nazývat τ-okolí.

4.5 Využití shlukové analýzy

Shluková analýza byla zevrubně popsána v sekci 2.3.2 včetně hrubého popisu algoritmu DBSCAN, který budeme využívat. Směrodatnými vlastnostmi, které vedly k této volbě je tvorba shluků libovolného tvaru, nezávislost na řazení databáze a vhodná parametrizace (ϵ, *minPts*). Důležité také jé, že není nutné znát předem počet shluků.

DBSCAN byl s úspěchem použit v práci [6] zabývající se tématem dolování periodických vzorů a také jako součást metody dolování *rojů* [28], což z něj dělá bezpečnou volbu pro naše potřeby.

Naše data jsou po transformacích popsaných v sekci 4.4 rozdělená do několika menších datasetů podle času. DBSCAN postupně aplikujeme na všechny tyto datasety. Vznikne tak databáze množin shluků seřazená podle časových okamžiků.

Následuje pseudokód algoritmu podle [8]:

```
DBSCAN (SetOfPoints, Eps, MinPts)
  // SetOfPoints is UNCLASSIFIED
  ClusterId := nextId(NOISE);
```

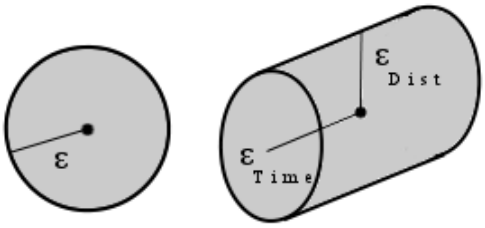
```
  FOR i FROM 1 TO SetOfPoints.size DO
    Point := SetOfPoints.get(i);
    IF Point.ClId = UNCLASSIFIED THEN
      IF ExpandCluster(SetOfPoints, Point, ClusterId, Eps, MinPts) THEN
        ClusterId := nextId(ClusterId)
      END IF
    END IF
  END FOR
END; // DBSCAN

ExpandCluster(SetOfPoints, Point, ClId, Eps, MinPts) : Boolean;
  seeds:=SetOfPoints.regionQuery(Point,Eps);
  IF seeds.size<MinPts THEN // no core point
    SetOfPoint.changeClId(Point,NOISE);
    RETURN False;
  ELSE   // all points in seeds are density-reachable from Point
    SetOfPoints.changeClIds(seeds,ClId);
    seeds.delete(Point);
    WHILE seeds <> Empty DO
      currentP := seeds.first();
      result := SetOfPoints.regionQuery(currentP, Eps);
      IF result.size >= MinPts THEN
        FOR i FROM 1 TO result.size DO
          resultP := result.get(i);
          IF resultP.ClId IN {UNCLASSIFIED, NOISE} THEN
            IF resultP.ClId = UNCLASSIFIED THEN
              seeds.append(resultP);
            END IF;
            SetOfPoints.changeClId(resultP,ClId);
          END IF; // UNCLASSIFIED or NOISE
        END FOR;
      END IF; // result.size >= MinPts
      seeds.delete(currentP);
    END WHILE; // seeds <> Empty
    RETURN True;
  END IF
END; // ExpandCluster
```

Návratovou hodnotou funkce `SetOfPoints.regionQuery(Point,Eps)` je seznam bodů v `EPS`-okolí bodu `Point` patřícího do `SetOfPoints`. Tato funkce bude pro každý bod zavolána přesně jednou. Využitím indexovací datové struktury obsahující předpočítané vzdálenosti můžeme složitost této funkce snížit na $O(\log n)$ a celkovou složitost algoritmu tak na $O(n \log n)$.

Vzhledem k tomu, že součástí datasetů, na které bude DBSCAN aplikován, může být více bodů náležící stejnému objektu, musíme z výpočtu množiny bodů v ϵ-okolí referenčního bodu vynechat ty se stejným identifikátorem jako má referenční bod.

Jednou z možností budoucího vývoje a potenciálního zlepšení výsledků našeho systému je vytvoření varianty algoritmu DBSCAN, ve které by v rámci ϵ-okolí bodu nebyl brán v potaz pouze prostor (tedy x, y a případně z souřadnice), ale současně by bylo rozšířeno i

do časové dimenze, tak, jak je nastíněno na obrázku 4.4. Takový přístup by mohl efektivně řešit problém časového posunu rozebraný v sekci 4.4.2.

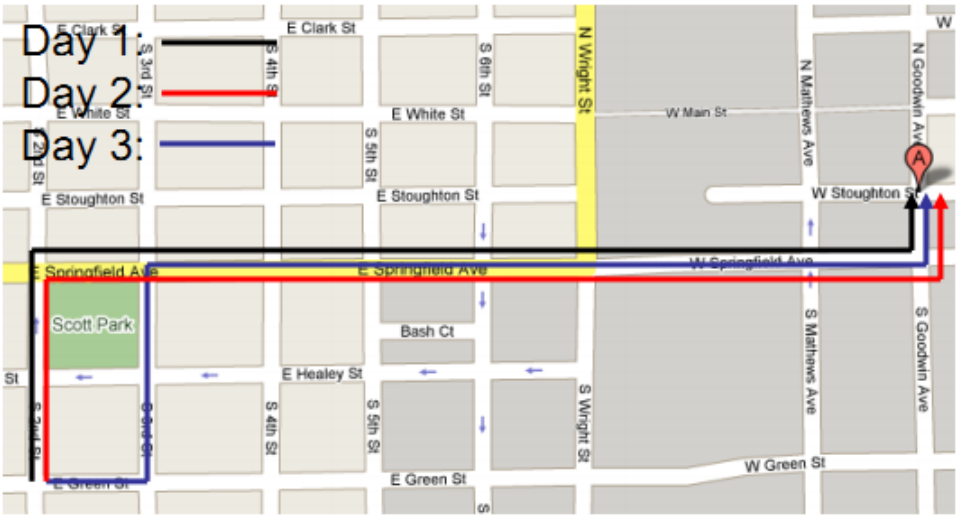


Obrázek 4.4: Alternativní přístup k výpočtu ϵ-okolí v algoritmu DBSCAN.

4.6 Dolování periodických pohybových vzorů

Jako definice periodického vzoru poslouží alternativní interpretace definice vzoru *roj*, definovaného v sekci 2.3.3. Ten je sice primárně určený k vyhledávání skupin objektů pohybujících se společně po dobu určitého časového intervalu, nicméně jeho vlastnosti jsou výhodné i z hlediska dolování periodických vzorů.

Stejně jako objekt, který se krátkodobě oddělí od skupiny, je stále intuitivně jejím členem, i drobné odchylky v jinak ustáleném periodicky se opakujícím chování neznamenají opuštění rutiny. Obrázek 4.5 ilustruje pohyb člověka po tři dny. Vyznačená cesta může být např. trasa z domova do práce a drobná odchylka v třetím dni může znamenat např. zvolení lehce odlišné cesty za účelem koupě snídaně. Přestože tyto tři trajektorie nejsou úplně totožné, intuitivně cítíme, že jsou součásti jedné rutiny. *Roj* na rozdíl od jiných pohybových vzorů neobsahuje podmínku náležitosti do skupiny pro jakékoliv dva časové okamžiky.



Obrázek 4.5: Periodický pohyb objektu. Převzato z [16]

Roj označuje časové okamžiky, kdy je množina rozdílných objektů blízko sebe. My sice

zkoumáme polohová data vždy pouze jednoho objektu, nicméně po rozdělení kompletní polohové historie objektu na časové úseky dle zvolené periody, je každý jeden z těchto úseků označen unikátním identifikátorem. Z hlediska *roje* k bodům z různých úseků tedy budeme přistupovat, jako kdyby byly vytvořené různými objekty. Po překrytí těchto úseků bude pak naše interpretace definice *roje* označovat časové okamžiky množiny úseků, kdy byl objekt přítomen na přibližně stejné poloze.

Uvažujme např. datovou sadu obsahující polohová data objektu vykazujícího periodické chování pořízená v intervalu sedmi dní. Zvolená perioda je jeden den. Výsledkem zpracování této datové sady bude *Roj* (O, T), kde $O = \{o_1, o_2, o_4, o_5, o_6\}$ a $T = \{t_1, \ldots, t_{20}, t_{30}, \ldots, t_{50}\}$. Standardní interpretace definice *roje* říká, že objekty o_1, o_2, o_4, o_5, o_6 cestovaly v časech $t_1, \ldots, t_{20}$ a $t_{20}, \ldots, t_{30}$ společně. Naproti tomu naše interpretace definice *roje* popisuje objekt, který se pohyboval ve dnech o_1, o_2, o_4, o_5, o_6 v časech $t_1, \ldots, t_{20}$ a $t_{20}, \ldots, t_{30}$ po stejné trajektorii.

4.6.1 Algoritmus objectGrowth

Abychom se vyhnuli dolování redundantních vzorů, rozšíříme definici *roje* ze sekce 2.3.3 o uzávěrové vlastnosti tak, jak je navrhováno v [28].

Definice 4.6.1. Uzavřený Roj(O, T): Roj je *objektově uzavřený*, pokud při fixní množině T již nemůže být množina O rozšířena (neexistuje O' takové, že (O', T) je roj a zároveň $O \subsetneq O'$). Podobně roj je *časově uzavřený* pokud při fixní množině O, množina T již nemůže být rozšířena (neexistuje T' takové, že (O, T') je roj a zároveň $T \subsetneq T'$). Pokud je roj zároveň objektově uzavřený i časově uzavřený, označujeme ho jako uzavřený roj.

K vyhledání periodických vzorů použijeme algoritmus *objectGrowth* představený v [28], jehož pseudokód je následující:

```
objectGrowth(O,T_max,o_last,min_o,min_t,|O_DB|,|T_DB|,C_DB)
{Apriori Prunning}
if |T_max| < min_t then
   return;
{Backward Prunning}
if BackwardPruning(o_last, O, T_max, C_DB) then
   forward_closure ← true;
   for o ← o_last + 1 to |O_DB| do
      O' ← O ∪ {o};
      T'_max ← GenerateMaxTimeset(o,o_last,T_max,C_DB);
      if |T_max| = |T'_max| then
         forward_closure ← false; {Forward Closure Checking}
      objectGrowth(O_new,T_new,o,min_o,min_t,|O_DB|,|T_DB|,C_DB);
   if forward_closure and |O| ≥ min_o then
      output pair (O,T) as a closed swarm;

Subroutine: BackwardPrunning(o_last,O,T_max,C_DB)
for ∀o ∉ O and o < o_last do
   if C_t(o) ⊆ C_t(O),∀t ∈ T_max then
      return false;
return true;
```

```
Subroutine: GenerateMaxTimeset(o,o_last,T_max,C_DB)
for ∀t ∈ T_max do
    if C_t(o) ∩ C_t(o_last) ≠ ∅ then
        T'_max ← T'_max ∪ t
return T'_max;
```

Algoritmus postupně v pořadí určeném pomocí metody slepého prohledávání do hloubky (angl. DFS – Depth First Search) prochází všechny podmnožiny O_{DB} a vytváří tak B-Tree datovou strukturu. Aby byla snížena velikost prohledávacího prostoru, jsou využita dvě pravidla prořezávání. Dále je použita metoda dopředné uzávěrové kontroly (angl. Forward Closure Checking), která ve stejném průchodu umožňuje označit *uzavřené roje* a oddělit je tak od *rojů* obyčejných. Algoritmus lze parametrizovat pomocí prahu min_o označujícího minimální počet objektů v *roji* a prahu min_t označujícího minimální počet časových okamžiků, v kterých musí být objekty pohromadě.

4.7 Klasifikace nové trajektorie

Při příchodu nové trajektorie do našeho systému se ji pokusíme zařadit do některého z dříve vydolovaných periodických vzorů. Jak vidíme ve schématu architektury systému (obrázek 4.1), nejprve provedeme stejné předzpracování a transformaci jako při zpracování celé historie uživatele. Tyto fáze procesu byly obecně popsány v sekci 2.2 a dále rozvinuty v sekcích 4.3 a 4.4.

Dalším krokem je průchod předem vytvořenou databází množin shluků po jednotlivých časových okamžicích. Shluková analýza byla dopodrobna popsána v sekci 2.3.2, námi vybraný algoritmus v sekci 2.3.2 a 4.5, kde se nachází i popis tvorby zmíněné databáze.

Bod trajektorie v určitém časovém okamžiku vždy porovnáme s body ve shlucích příslušejících ke stejnému časovému okamžiku. Toto porovnání bude probíhat na základě ϵ-okolí. Pokud se bude některý z bodů některého shluku nacházet v ϵ-okolí příslušného bodu trajektorie, tento shluk si vždy označíme. Výsledkem této fáze je nová databáze množin shluků taková, že body trajektorie by na základě vzdálenosti mohly být členy těchto shluků.

Za pomoci takto vytvořené nové databáze množnin shluků a připravených periodických vzorů získaných pomocí metody popsané v sekci 4.6.1 provedeme samotnou klasifikaci trajektorie. Analogicky k metodě *objectGrowth* postupně projdeme stromovou strukturu B-Tree obsahující *uzavřené roje* v pořadí určeném metodou slepého prohledávání do hloubky. Vždy porovnáme *podobnost* (viz dále) zpracovávané trajektorie a daného *uzavřeného roje*. Po průchodu stromem vyhodnotíme hodnoty *podobnosti* všech vzorů a vybereme ten s nejvyšší mírou *podobnosti*. Pokud žádný ze vzorů nebude vykazovat *podobnost* vyšší než práh s_{min}, označíme trajektorii jako *nestandardní*.

Podobnost *uzavřeného roje* (O, T) a zpracovávané trajektorie vypočítáme následovně: Postupně projdeme všechny časové okamžiky patřící do množiny T. Ty nám poslouží jako index do databáze množin shluků. Pokud je množina objektů O podmnožinou některého z množiny shluků dané příslušným časovým okamžikem, tento časový okamžik si označíme. Mírou podobnosti je pak poměr označených časových okamžiků vůči $|T|$.

Kapitola 5

Implementace navrženého systému

Tato kapitola se zabývá implementací systému navrženého v minulé kapitole. Nejprve uvedeme použité nástroje, knihovny a programovací jazyk. Následně popíšeme formát vstupních dat použitý ve vybrané datové sadě. Jsou popsány dva přístupy k dekompozici úlohy, na jejich základě je vytvořen diagram tříd a vysvětlena samotná implementace řešení úlohy. Poslední podsekce je věnována vizualizaci polohových dat.

5.1 Volba vývojového prostředí a knihoven

S přihlédnutím ke zkušenostem byl k implementaci zvolen objektově orientovaný programovací jazyk C++ a vývojové prostředí Microsoft Visual Studio 2010, vyvinuté pro operační systém Microsoft Windows. Nicméně implementace je s malými změnami přenositelná i na ostatní běžně používané operační systémy.

Abychom co nejvíce zvýšili rychlost výpočtů shlukové analýzy, která je z hlediska nároků na výpočetní prostředky v našem systému nejkritičtější, byla využita knihovna *Boost*[1], která je jedinou externí nestandardní knihovnou v implementaci.

5.2 Vstupní data

Implementace počítá s daty v PLT formátu, kde každý PLT soubor obsahuje jednu trajektorii uživatele a je pojmenovaný jejím počátečním časem. Data každého uživatele jsou uložena v oddělené složce.

Implementace předpokládá vstupní data v PLT formátu, kde každý PLT soubor obsahuje jednu trajektorii uživatele a je pojmenovaný jejím počátečním časem. data každého uživatele jsou uložená v oddělené složce. Soubor v PLT formátu vždy obsahuje 6 řádků hlavičkových dat, která jsou pro nás nepotřebná a která budeme ignorovat. Jednotlivé body jsou popsané na dalších, datových řádcích. Každý bod je uveden na samostatném řádku a rozdělen do několika polí, oddělených čárkou.

Datový řádek PLT formátu

- Pole 1 – Zeměpisná šířka ve stupních.
- Pole 2 – Zeměpisná délka ve stupních.

[1] http://www.boost.org/

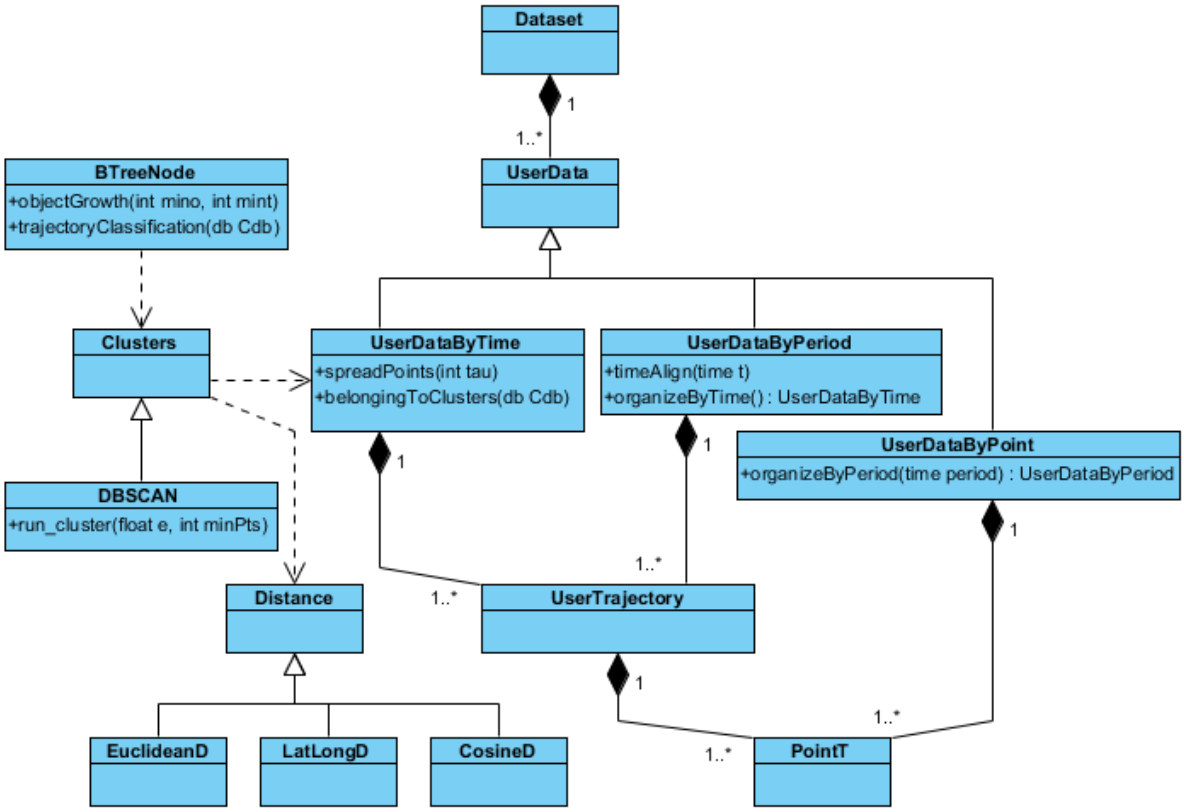
- Pole 3 – Nastaveno na ’0’ pro všechny body v této datové sadě.
- Pole 4 – Nadmořská výška ve stopách (Nastaveno na -777, pokud je neplatná).
- Pole 5 – Datum - počet dní (s desetinnou částí), které uplynuly od 30.12.1899.
- Pole 6 – Datum jako řetězec (GMT).
- Pole 7 – Čas jako řetězec (GMT).

Příklad záznamu dvou navazujících bodů:

```
39.906631,116.385564,0,492,40097.5864583333,2009-10-11,14:04:30
39.906554,116.385625,0,492,40097.5865162037,2009-10-11,14:04:35
```

5.3 Organizace tříd

Na problém dekompozice úlohy lze nazírat dvěma pohledy. Doposud jsme aplikovali přístup, kdy jsme na úlohu nahlíželi z hlediska funkčnosti. Rozdělili jsme výpočet na dílčí úlohy a těm pak přidělili data. Druhá možnost, která je z hlediska objektově orientovaného přístupu intuitivnější a výhodnější, je dekompozice a řešení úlohy z pohledu dat. Ta spočívá ve vytvoření hierarchie datových struktur (tříd) a pak z rozhodnutí, jak s těmito daty pracovat. Zjednodušený diagram tříd naší implementace lze vidět na obrázku 5.1



Obrázek 5.1: Diagram tříd.

5.3.1 Datová hierarchie

Nejmenším stavebním prvkem datové sady je bod v prostoru označující polohu objektu v určitém čase. Třída `PointT` tak intuitivně obsahuje zeměpisnou šířku, zeměpisnou délku, nadmořskou výšku, což jsou souřadnice, z kterých se skládá poloha. Dále je pak její součástí identifikátor objektu, jehož poloha byla zaznamenána, a čas, kdy se tak stalo.

Body jsou dále v libovolném počtu řazeny do trajektorií, které v naší implementaci reprezentuje třída `UserTrajectory`. Seskupené trajektorie představují uživatelská data v podobě třídy `UserData`. Podle způsobu reprezentace uživatelských dat jsou odvozeny potomci této třídy `UserDataByTime`, `UserDataByPeriod` a `UserDataByPoint`. Poslední jménovaná třída vynechává trajektorie a je sestavena pouze ze sekvence bodů. Data jednotlivých uživatelů jsou pak součástí třídy `Dataset`, která je zastřešuje. Vzhledem k tomu, že budeme pracovat vždy pouze s jedním uživatelem, je poslední zmíněná třída spíše symbolická, připravená k případnému budoucímu vývoji zabývajícímu se studiem meziuživatelských vlivů a vztahů.

Typ reprezentace uživatelských dat `UserDataByTime` je společně s třídou `Distance`, představující použitý typ vzdálenosti, základem pro vytvoření shluků bodů (třída `Clusters`). Shluky jsou pak zase esenciální pro vybudování stromové struktury z objektů třídy `BTreeNode`. Tato stromová struktura uchovává kýžené periodické vzory (*roje*).

5.3.2 Funkční dekompozice a metody tříd

V předchozí podsekci jsme si ujasnili, jak budou data reprezentována a jaké jsou mezi nimi vztahy a závislosti. Nyní vytvoříme metody, s jejichž pomocí nad zmiňovanými daty budeme provádět operace simulující proces popsaný v předchozí kapitole.

Správa dat z pohledu tříd

Třídy pracující přímo s datovou sadou musí obsahovat metody pro načítání dat ze souborů a jejich uložení v požadovaném formátu. Tyto operace je vhodné realizovat již v konstruktorech tříd. Při vytvoření např. objektu třídy `UserData` tak mohou být automaticky kaskádově volány konstruktory hierarchicky nižších tříd. Další možnost je načítat data externě a pak je objektu třídy přiřadit. V diagramu tříd (obrázek 5.1) vidíme, že některé třídy jsou hierarchicky na stejné úrovni. Představují varianty reprezentace stejných dat, které mohou být výhodné pro různé typy operací. Je tedy vhodné implementovat metody pro změnu reprezentace dat.

Jednotlivé operace nad daty implementující proces popsaný schématem architektury systému (obrázek 4.1) vždy implementujeme jako metodu třídy, která z hlediska operace představuje nejvhodnější reprezentaci dat.

Zpracování historie uživatele

Pro načtení dat ze souboru při zpracování kompletní historie uživatele nejprve zvolíme nejjednoduší reprezentaci uživatelských dat, třídu `UserDataByPoint`. Vždy zkontrolujeme časovou vzdálenost dvou navazujících bodů a případně zareagujeme adekvátní metodou předzpracování (vyhození poškozených dat, lineární interpolace chybějících dat apod.). Následně pomocí metody `organizeByPeriod` rozdělíme data na úseky o délce zvolené periody. Současně jim přiřadíme identifikátor úseku. Vznikne reprezentace uživatelských dat v podobě třídy `UserDataByPeriod`. Nyní zarovnáme a sesynchronizujeme časové okamžiky

překrytých úseků bodů (metoda `timeAlign`). Podle nich znovu změníme reprezentaci uživatelských dat na `UserDataByTime`. O rozkopírování bodů v časovém τ-okolí za účelem kompenzace časových posunů a tedy vyšší úspěšnosti shlukové analýzy se stará metoda `spreadPoints`.

Shlukovou analýzu implementujeme jako metodu třídy `Clusters`, resp. jejího potomka `DBSCAN`. Ta pomocí opakovaného volání metody `runCluster` (pseudokód v sekci 4.5) a uživatelských dat, která jsou součástí objektu třídy `UserDataByTime` vytvoří databázi množin shluků pro jednotlivé časové okamžiky. Pro urychlení výpočtu této metody si předem předpočítáme vzdálenosti bodů a uložíme je do objektu třídy `Distance`, resp. do jednoho z jeho potomků.

Metoda `objectGrowth` s využitím databáze množin shluků vytváří stromovou strukturu z objektů třídy `BTreeNode` a zároveň tak v datech vyhledává pohybový vzor *roj*. Pseudokód tohoto procesu lze nalézt v sekci 4.6.1.

Klasifikace trajektorie

Nově přijatá uživatelská trajektorie projde standardním předzpracováním a transformací. Vlivem rozkopírování bodů do časového τ-okolí je změněna její reprezentace na objekt třídy `UserDataByTime`, protože v každém časovém okamžiku bude obsaženo více bodů. Proto je operace ustanovení příslušnosti ke shlukům `belongingToClusters` implementována jako metoda této třídy. Klasifikace do vzorů je pak realizována metodou `trajectoryClassification` třídy `BTreeNode`.

Doplňkové metody

Součástí implementace jsou doplňkové metody umožňující tisk informací o datech do konzole, testování funkčnosti metod, umělé generování polohových dat, řazení polohových dat dle data či času a export polohových dat a vzorů do souborů formátu `JSON`, který lze následně využít k vizualizaci.

5.4 Vizualizace pomocí Google Maps

Google Maps je webová mapová služba od společnosti Google. Nabízí interaktivní dopravní či satelitní mapu. Mezi funkční prvky patří přibližování a oddalování mapy, pohyb mapy pomocí myši, plánování cest aj.

Společnost Google bezplatně k nekomerčnímu využití nabízí *Google Maps JavaScript API v3*, které nám poslouží k vizualizaci hrubých dat i výsledků. Umožňuje stáhnout interaktivní mapu do prohlížeče a přímo, či na základě událostí, do ní zakreslovat značky, křivky, polygony atd. Ty lze uživatelsky přizpůsobit (různé animace, ikony, barvy apod.).

Abychom si mohli lépe představit rozložení hrubých polohových dat, shluků z nich vytvořených a alespoň přibližou podobu vzorů, byla vytvořena jednoduchá aplikace za pomoci programovacího jazyka *javascript*. Její součástí je interaktivní google mapa a rozhraní pro vkládání polohových dat ze souborů. Lze zvolit vykreslení dat v podobě značek nebo body spojit do křivek. V případě vykreslování shluků nebo vzorů je můžeme nechat vykreslit všechny najednou nebo vykreslování krokovat.

Data budeme do aplikace přivádět pomocí rozhraní *JavaScript File API*. Pro jednoduchost a snadnou srozumitelnost využijeme formát JSON (JavaScript Object Notation).

Předzpracovaná uživatelská data jsou rozdělena do menších úseků, kde každý úsek je opatřen identifikátorem a seznamem polohových bodů opatřených časovým razítkem. Zápis v JSON formátu vypadá následovně:

```
{
  "userData": [
    {"id": int,
     "points:"[
       {"timestamp:" double,
        "longitude:" double,
        "latitude:" double},
        ...]
    },
    ...]
}
```

Databáze množin shluků připravená k vizualizaci je seřazená podle časových okamžiků. V každém časovém okamžiku je přítomno určité množství shluků bodů.

```
{
  "timestamps": [
    {"timestamp": double,
     "clusters:"[
       {"points":[
         {"timestamp:" double,
          "longitude:" double,
          "latitude:" double},
          ...]
       },
       ...]
    },
    ...]
}
```

Vizualizace periodických vzorů je problematická. Časové okamžiky od sebe mohou být různě daleko a každý časový okamžik obsahuje shluk bodů. Rozhodl jsem se tedy každý shluk reprezentovat jedním referenčním bodem, jehož poloha je střed shluku.

```
{
  "swarms": [
    {"timestamp": double,
     "points:"[
       {"timestamp:" double,
        "longitude:" double,
        "latitude:" double},
        ...]
    },
    ...]
}
```

Kapitola 6

Volba parametrů, experimenty a jejich výsledky

V první části této kapitoly si rozebereme parametrizovatelné metody našeho systému. Vysvětlíme, jaký vliv mají parametry na zpracování uživatelské polohové historie a proč tomu tak je. Otestujeme jednotlivé části tohoto procesu a provedeme analýzu výsledků.

Druhá část kapitoly se věnuje klasifikaci příchozí trajektorie do předem vytvořených vzorů za pomoci databáze shluků. Klasifikaci budeme testovat na sadách vzorů a shluků vytvořených různými uživateli.

6.1 Parametrizovatelné metody

Část metod, které dohromady implementují proces zpracování kompletní historie uživatelských dat je parametrizovatelná. Volba těchto parametrů může mít vliv na kvalitu a množství výsledků, na dobu výpočtu atd.

6.1.1 Metody předzpracování

V této sekci budeme diskutovat vliv parametrů metod předzpracování na průběh shlukové analýzy a nepřímo tedy i na dolování vzorů. Experimenty v této podsekci byly prováděny s následujícími parametry: $t_{vz} = 5\,\text{s}$, $P = 1\,\text{den}$, $minPts = 3$, $\epsilon = 30\,\text{m}$.

Vliv na výsledky shlukové analýzy budeme měřit na základě těchto atributů:

- $N[\text{t}]$ – Počet časových okamžiků obsahujících nějaké shluky.
- $N[\text{cl/t}]$ – Průměrný počet shluků na časový okamžik.
- L – Průměrná velikost shluku.

Volba periody

Jak bylo uvedeno v sekci 4.4.1, periodu P budeme volit ručně na základě přirozených přírodních cyklů. Nejkratší časovou jednotkou spadající do této kategorie je jeden den. Zvolení nejmenší možné časové jednotky, dostačující k vytvoření přirozeného periodického chování, jako periodu znamená rozdělení datové sady na maximální možný počet úseků a tedy nejlepší podmínky pro detekci periodického chování. Proto budeme v našich experimentech téměř výhradně pracovat s periodou $P = 1\,\text{den}$.

Volba hodnoty časového kroku

Jak je vysvětleno v sekci 4.4.2, polohová data se vzorkovací periodou t_{vz} jsou zarovnávána podle pevného, *uměle zvoleného* časového kroku t_{step}. Existují tři možnosti volby hodnoty t_{step} ve vztahu k t_{vz}:

1. $t_{step} < t_{vz}$ – Tato volba je nesmyslná, protože bychom získali pouze prázdné časové okamžiky, ke kterým by nepříslušely žádné polohové body, popř. pouze kopie bodů, které bychom do těchto časových okamžiků nakopírovali v další fázi.

2. $t_{step} = t_{vz}$ – Pouze synchronizace časových okamžiků jednotlivých překrytých časových úseků. V každém takovém úseku jsou všechny polohové body v čase posunuty jedním (podle polohy referenčního bodu) směrem o hodnotu menší, než je $\frac{t_{vz}}{2}$. Tato volba nijak nepomáhá řešení problému časových posunů, popsaného v sekci 4.4.2.

3. $t_{step} > t_{vz}$ – Při této volbě vznikne stav, kdy v každém (za předpokladu úplných dat) nově vzniklém časovém okamžiku bude několik polohových bodů.

Volba hodnoty t_{step} má přímý vliv na konečný počet časových okamžiků, kterých po této operaci bude $\frac{P}{t_{step}}$, kde jako P označujeme zvolenou periodu.

V tabulce 6.1 vidíme výsledky experimentu zabývajícího se vlivem volby t_{step} na výsledky shlukové analýzy, prováděného na datové sadě uživatele s identifikátorem 002, obsahující trajektorie z 118 dní. Parametr τ je v tomto případě nastaven na 0.

t_{step}[s]	N[t]	N[cl/t]	L
10	3974	1,68	6,79
20	2081	1,69	6,97
30	1405	1,69	7,25
60	731	1,7	7,6

Tabulka 6.1: Vliv parametru t_{step} na výsledky shlukové analýzy.

Přesná čísla v tabulce nejsou příliš podstatná, protože se odvíjejí od délky a kvality datové sady, nicméně lze z nich vypozorovat míru růstu či poklesu hodnot atributů shluků. Se zvyšováním hodnoty t_{step} a tedy celkovým snížením počtu časových okamžiků logicky klesá i počet časových okamžiků, které obsahují nějaké shluky. Tento pokles je nižší, než lineární, protože snížení počtu časových okamžiků neznamená snížení počtu polohových bodů. Ty jsou rozprostřeny mezi zbývající časové okamžiky a mohou tak vytvořit shluky v časových okamžicích, v kterých to při menším časovém kroku t_{step} nebylo možné. Počet shluků na časový okamžik se příliš nemění, ale jejich velikost stoupá, jak se stále více polohových bodů shromažďuje v menším počtu časových okamžiků. To může do jisté míry pomoci se vypořádat s možným časovým posunem trajektorií (viz sekce 4.4.2), nicméně při příliš velkém časovém kroku t_{step} automaticky dochází k příliš velkému časovému posunu jednotlivých bodů a následovnému zkreslení výsledků.

Parametr τ

Parametr τ ovlivňuje velikost τ-okolí, což je časová vzdálenost, v rámci které budou rozkopírovány body mezi sousedy referenčního bodu. Tato metoda podle [6] výrazně přispívá k překonání překážky v podobě časových posunů trajektorií.

S parametrem τ je třeba nakládat opatrně, protože při hodnotě $\tau > 0$ dochází k podstatnému navýšení objemu dat, s kterými bude muset následující shluková analýza a algoritmus pro dolování vzorů pracovat. Po replikaci bude datová sada určená ke zpracování obsahovat $(2 \cdot \tau + 1) \cdot N$ bodů. (N zde značí původní počet bodů).

V tabulce 6.2 vidíme výsledky experimentu zabývajícího se vlivem volby τ na výsledky shlukové analýzy, prováděného na datové sadě uživatele s identifikátorem 028 obsahující trajektorie z 60 dní, při $t_{step} = 10$.

τ	**N[t]**	**N[cl/t]**	**L**
0	1622	1,48	7,78
1	1715	1,81	7,78
2	1770	2,07	7,76
3	1812	2,23	7,86
5	1904	2,41	7,99

Tabulka 6.2: Vliv parametru τ na výsledky shlukové analýzy.

V tabulce je vidět, jak má replikace bodů v τ-okolí příznivý vliv na tvorbu shluků. Vzniká tak více časových okamžiků obsahujících shluky a stoupá i průměrný počet shluků na jeden časový okamžik.

6.1.2 Shlukovací algoritmus DBSCAN

U shlukovacího algoritmu lze volitelně nastavit parametr $MinPts$ určující minimální počet bodů, které mohou tvořit shluk, a parametr ϵ, který vymezuje oblast ϵ-okolí, ve které se vyhledávají další kandidátní body.

Parametr ϵ

Shluky jsou tvořeny na základě vzdálenosti bodů. Ta je volitelná pomocí parametru ϵ. Implementovány byly Euklidovská vzdálenost a vzdálenost v metrech s využitím vzorce Haversine [36]. Vzhledem k formátu vstupních dat a přesnosti uvedených metrik budeme v experimentech využívat výhradně vzdálenost v metrech.

Experiment, který shrnuje tabulka 6.3 byl proveden s parametry $t_{vz} = 5\,\text{s}$, $P = 1\,\text{den}$, $minPts = 3$, $t_{step} = 10$, $\tau = 2$ na datové sadě uživatele s identifikátorem 028 obsahující trajektorie ze 60 dní.

ϵ	**N[t]**	**N[cl/t]**	**L**
5	1552	2,17	5,76
10	1610	2,14	6,89
20	1701	2,11	7,62
30	1770	2,07	7,76
60	1848	1,94	8,29
200	2003	1,76	9,05

Tabulka 6.3: Vliv parametru ϵ na výsledky shlukové analýzy.

Jak lze vyčíst z tabulky, zvyšování ϵ-okolí má za následek zvýšení počtu časových okamžiků obsahujících nějaké shluky a také zvýšení průměrné velikosti shluku. Větší prohle-

dávaná oblast samozřejmě obsahuje více bodů, nicméně strmost tohoto nárustu postupně klesá, jak je prohledáváno okolí dále od center shluků. Neuvážené zvyšování hodnoty ϵ může mít nežádoucí účinky, jako je spojování shluků apod., viz v tabulce viditelná klesající hodnota počtu shluků na časový okamžik.

Parametr *minPts*

Standardním účelem volby hodnoty parametru *minPts* je stanovení dolní hranice velikosti shluku. V našem případě tato volba přímo vypovídá o tom, kolikrát se v různých úsecích o zvolené periodě musí *v daném časovém okamžiku* poloha objektu opakovat, abychom to označili jako *periodický jev*. Je zbytečné tvořit shluky o velikosti menší než je požadovaná hodnota periodicity.

Parametr *minPts* nelze zvolit pevně pro všechny uživatelské datové sady. Je třeba přihlédnout k jejich délce. Je samozřejmé, že např. při uživatelské datové sadě o délce jednoho roku nalezneme více dní (předpokládáme $P = 1\,\text{den}$), kdy byl objekt na stejném místě, než kdybychom zpracovávali uživatelskou datovou sadu o délce jednoho týdne. Nabízí se tedy volba parametru *minPts* na základě požadovaného percentuálního zastoupení periodického chování v datové sadě. Pokud bychom např. měli datovou sadu o délce 10 dní a požadovaná periodicita by byla 40 %, parametr *minPts* bychom nastavili na hodnotu 4. Ani tento přístup k volbě parametru ale nemusí být vždy správný. Při velmi dlouhé uživatelské historii se dá předpokládat, že periodická chování budou postupně vznikat a zanikat. Může dojít ke změně návyků, navázání nové známosti, školáci mají jiné rutiny během školního roku a o prázdninách apod. Při takto globálně zvoleném *minPts* by nám mohlo velké množství periodického chování uniknout.

Rozdělíme si typ periodického chování na dvě skupiny a k jejich dolování budeme přistupovat odlišným způsobem.

1. Lokální periodické chování – Periodické chování, které se objevuje pouze v rámci určitého časového intervalu, např. periodické chování na dovolené nebo při služební cestě. Při dolování lokálního periodického chování si uživatelskou historii rozdělíme na menší části a pokusíme se ho detekovat postupně, v rámci těchto menších datových sad.

2. Globální periodické chování – Chování, které považujeme za dlouhodobě stabilní. Do této skupiny spadá např. cesta z domova do zaměstnání. Detekce bude probíhat na celé uživatelské historii.

6.1.3 Metoda objectGrowth

Metoda objectGrowth obsahuje implementaci algoritmu pro vyhledávání *uzavřených rojů* (O, T), jehož pseudokód je popsaný v sekci 4.6.1. Možné velikosti množin O a T jsou přímo závislé na parametrech min_o a min_t.

Parametr min_o

Analogicky k parametru *minPts*, parametr min_o představuje dolní hranici možného počtu objektů k vytvoření vzoru. Vymezuje tedy, v kolika různých úsecích o zvolené periodě se musí (*ve všech časových okamžicích patřících do množiny* T) poloha objektu opakovat, abychom toto chování označili za periodický vzor.

Parametry $minPts$ a min_o spolu přímo souvisí a tak by jejich volba měla být koordinována. Jejich vztah má následující vliv na dolování periodických vzorů:

1. $min_o > minPts$ – Metoda objectGrowth bude zbytečně zpracovávat shluky o velikosti $minPts$ až min_o, přestože v nich není možné nalézt množinu objektů O, která by měla požadovanou velikost.

2. $min_o < minPts$ – Nedochází k žádnému nadbytečnému zpracování shluků, nicméně pokud považujeme hodnotu min_o dostatečnou k popisu periodického vzoru, měli bychom brát v potaz i shluky o velikosti min_o až $minPts$. Ty ale nebyly vlivem příliš vysoké hodnoty $minPts$ vytvořeny.

3. $min_o = minPts$ – Nedochází k žádnému nadbytečnému zpracování shluků a máme k dispozici všechny shluky požadované velikosti. Pochopitelně budeme v experimentech vždy používat tento vztah.

Parametr min_t

Parametr min_t ovlivňuje časovou délku vzoru. Vymezuje, v kolika časových okamžících se musí vyskytnout periodický jev, tedy shluk bodů, kterého jsou součástí všechny objekty patřící do množiny O, abychom toto chování označili za periodický vzor.

Při volbě tohoto parametru bychom měli přihlédnout ke granularitě dat, konkrétně pak k velikosti vzorkovací periody t_{vz} a velikosti umělého časového kroku t_{step}. Ty diskretizují jinak spojitý čas do určitého množství časových okamžiků. Např. 1 min je při $t_{vz} = 5$ sec rozdělena do 12 časových okamžiků, ale při $t_{vz} = 10$ s pouze do 6 časových okamžiků. Pokud v tuto dobu probíhalo periodické chování, bylo zachyceno právě v tomto množství časových okamžiků. Proto je při vyšších hodnotách t_{vz} a t_{step} vhodné volit nižší hodnotu min_t a naopak.

Experimentální ověření vlivu min_o a min_t

Experimentální ověření proběhlo na datové sadě uživatele s identifikátorem 071, obsahující trajektorie ze 60 dní, při parametrech $t_{vz} = 5$ sec, $P = 1$ den, $t_{step} = 10$, $\tau = 2$, $\epsilon = 30$, $minPts = min_o$.

min_o	min_t	**N[roj]**	min_o	min_t	**N[roj]**	min_o	min_t	**N[roj]**
3	2	636	3	5	558	3	10	425
4	2	518	4	5	459	4	10	346
5	2	359	5	5	318	5	10	229
6	2	198	6	5	187	6	10	126

Tabulka 6.4: Vliv parametrů min_o a min_t na počet vydolovaných periodických vzorů.

Tabulka 6.4 ukazuje vliv parametrů min_o a min_t na počet vydolovaných periodických vzorů (*rojů*) – viz sloupec N[roj].

6.2 Klasifikace nové trajektorie

Druhá část systému spočívá v klasifikaci nově příchozí trajektorie do připravených vzorů s pomocí databáze množin shluků. Rozebereme faktory, které úspěšnost klasifikace ovlivňují

a provedeme experimentální ověření účinnosti klasifikátoru.

6.2.1 Faktory ovlivňující úspěšnost

Existuje několik faktorů, které mají vliv na úspěšnost klasifikace. Některé můžeme alespoň do určité míry ovlivnit, některé ne. Uvedeme si ty hlavní z nich.

Periodičnost chování

Jak bylo zmíněno v sekci 4.2.1, nelze očekávát úspěšnou detekci periodických vzorů a následnou klasifikaci u záznamu chování, které periodické není. Někteří lidé žijí v rutině více než jiní a někteří z hlediska polohových systémů nežijí v rutině vůbec. Někteří svoje chování mohou opakovat s jinou periodou než s tou, kterou jsme zvolili. Vzhledem k tomu, že zvolená datová sada není v tomto směru nijak označena, neznámá míra periodičnosti do experimentů zanáší velkou míru neurčitosti.

Nestandardní trajektorie

Do klasifikačního systému může přijít z hlediska periodicity nestandardní trajektorie, např. záznam trasy pořízené v rámci jednodenního výletu. V takovém případě samozřejmě klasifikátor v klasifikaci do některého z periodických vzorů selže. Toto není považováno za neúspěch klasifikátoru. Možným případem reálného využití klasifikátoru může naopak být právě detekce nestandardního chování. Bohužel bez preemptivních znalostí testovací datové sady nelze triviálně odlišit klasifikátorem správné označení nestandardní trajektorie od neúspěchu klasifikátoru klasifikovat trajektorií spadající do některého periodického vzoru.

Velikost a kvalita datové sady

Rozhodující vliv na klasifikaci má velikost a kvalita datové sady určené k vytvoření databáze množiny shluků a vyhledávání periodických vzorů, na kterých je klasifikace založena. Pozitivně klasifikaci ovlivňuje délka datové sady, kterou v rámci našich experimentů měříme ve dnech, protože umožňuje vytvořit delší a přesnější periodické vzory. Delší databáze nicméně nezaručuje kvalitnější data. Pozorovaná množina uživatelů z jejichž dat je vytvořena datová sada GeoLife GPS Trajectories nemá zapnuté GPS zařízení neustále. Zaznamenávané trajektorie jsou různě dlouhé. Např. tři velmi krátké trajektorie pořízené ve třech odlišných dnech mohou mít menší informační hodnotu než jedna dlouhá trajektorie mapující historii celého jediného dne.

Zvolené parametry metod

Vliv parametrů na proces vytváření databáze vzorů a množin shluků jsme si rozebrali v minulé sekci 6.1. Parametry metod předzpracování t_{step} a τ mají přímý vliv i na klasifikaci, protože příchozí trajektorie před samotnou klasifikací prochází stejným procesem předzpracování. Parametry ϵ, $minPts$, min_o, min_t pak ovlivňují klasifikaci nepřímo, vlivem na výsledky shlukové analýzy a algoritmu pro vyhledávání vzorů.

U našeho způsobu klasifikace lze navíc zvolit hodnotu parametru s_{min} označujícího minimální míru podobnosti trajektorie vůči vzoru dostačující na to, aby byla trajektorie do tohoto vzoru klasifikována. Výpočet podobnosti je popsán v sekci 4.7.

6.2.2 Experimentální ověření úspěšnosti klasifikace

Vzhledem k limitovanému množství dostupných dat se nebudeme striktně řídit obvyklým postupem k otestování úspěšnosti klasifikátoru. Ten předpokládá rozdělení dat na tréninková, sloužící k trénování klasifikátoru, a testovací, na kterých se ověřuje úspěšnost klasifikátoru. Namísto toho vždy odebereme jednu uživatelskou trajektorii a na zbylých provedeme trénink klasifikátoru, tedy vytvoření databáze vzorů a množin shluků. Dříve odebraná trajektorie poslouží jako testovací. Po zaznamenání úspěšnosti klasifikace trajektorie ji vrátíme do datové sady, odebereme jinou a postup opakujeme, dokud klasifikátor nebyl otestován na dostatečně velkém vzorku dat.

Databáze vzorů a množin shluků byla vytvořena s parametry $t_{vz} = 5\,\text{s}$, $P = 1\,\text{den}$, $t_{step} = 10$, $\tau = 2$, $\epsilon = 30$, $minPts = min_o = 3$, $min_t = 4$, $s_{min} = 30$. Úspěšnost klasifikátoru budeme testovat vždy na 10 ti náhodně vybraných trajektoriích z uživatelské datové sady. Jako celkovou úspěšnost klasifikátoru budeme uvažovat poměr úspěšně klasifikovaných trajektorií vůči celkovému počtu testovaných trajektorií.

Uživatel	**Délka datasetu[den]**	**Úspěšnost[%]**
012	57	50
071	60	60
115	123	60
119	44	40
179	46	40

Tabulka 6.5: Úspěšnost klasifikace.

Tabulka 6.5 ukazuje celkovou úspěšnost klasifikátoru u různých uživatelských datových sad. Experimenty ukázaly, že není výjimkou, aby trajektorie byla klasifikována s vysokou pravděpodobnosti do několika periodických vzorů najednou. Vydolované vzory mohou být v poměru k příchozí trajektorii krátké a mohou tak zachycovat jen část periodického chování a příchozí trajektorie může tyto podvzory spojovat. Druhou možností interpretace této několikanásobné klasifikace je, že trajektorie skutečně patří do více periodických vzorů. Pokud např. existují dva vzory, kde první popisuje trasu z domova do práce a druhý trasu z domova do obchodu, trajektorie zaznamenávající celodenní pohyb může spadat do obou těchto vzorů.

Kapitola 7

Závěr

V práci jsou popsány obecně používané techniky pro dolování v datech rozšířené o pohled na jejich využití v rámci dat pohybujících se objektů. V první kapitole jsou popsány jednotlivé fáze procesu získávání znalostí z databází ve vztahu k polohovým datům. Větší prostor je věnován shlukové analýze a vyhledávání pohybových vzorů.

Druhá část představuje úvod do problematiky rozpoznávání aktivit a cílů a přehled dosavadního vývoje na toto téma. Jsou zde představeny některé práce věnující se odvozování a rozpoznávání aktivit z polohových dat. Vzhledem k tomu, že drtivá většina těchto studii je z posledních let, tento přehled nastiňuje aktuální trendy v dané oblasti a ukazuje intenzivitu s jakou je toto atraktivní téma zkoumáno.

Byl vytvořen návrh systému pro rozpoznávání rutinního chování lidí využívající shlukovou analýzu a následné dolování vzorů, tedy technik popsaných v první kapitole. Návrh využívá některé principy a metody zejména z oblasti dolování frekventovaných vzorů popsané v kapitole druhé. Součást návrhu je také klasifikace nových trajektorií do dříve rozpoznaných periodických vzorů.

Tento systém by implementován v programovacím jazyce C++. Čtvrtá kapitola obsahuje rozbor této implementace obsahující úvahu nad možnými pohledy na dekompozici problému a dále pak popis tříd a metod implementujících navržený systém. V krátkosti je popsána aplikace sloužící k vizualizaci uživatelských polohových dat, jejich shluků a periodických vzoru.

Poslední kapitola je věnována analýze vlivu parametrů na dobu výpočtu a kvalitu výsledků. Různá nastavení parametrů byla podrobena experimentům. Výsledky těchto experimentů byly diskutovány a na jejich základě byla předložena doporučení k optimální volbě jejich hodnot. Součástí této kapitoly je také experimentální ověření úspěšnosti klasifikace a rozbor faktorů, které na ní mají vliv.

Shluková analýza je běžně využívaná technika při získávání znalostí z datových sad. Její použití jako součást dolování vzorů bylo popsáno v [6] a [28]. Tyto dvě práce mě inspirovaly k prozkoumání možnosti použít vzor *roj* k dolování periodických dat. Jak experimenty ukazují, jeho volná definice, která není podmíněna bezprostřední návazností shluků objektů v jakýchkoliv dvou časových okamžicích, umožňuje jinak velmi obtížnou detekci periodického chování, které se vyznačuje odchylkami, pusuny v čase atd.

Vzhledem k nízkému množství prací na téma klasifikace trajektorii na základě periodických vzorů, navíc založených na GPS trajektoriích lidí, je obtížné porovnat výsledky a určit úspěšnost přístupu popsaného v této práci. Extrakcí lidských denních rutin se zabývají práce [7] a [9]. Ty ale nevyužívají GPS souřadnice, pouze obecné základní stavy objektů („v práci“, „doma“, „bez signálu“ atd.). Vyspělou prací na téma dolování frekventovaných

vzorů je [29], nicméně pracuje s polohovými daty zvířat, která vykazují periodické chování o mnohem delší periodě, než s kterou bylo experimentováno v této práci. Nejblíže k přístupu popsanému v této práci má studie [6]. Ta je ale testována na uměle vygenerovaných datech, která netrpí problémy datových sad obsahující reálná data.

Budoucí vývoj práce by bylo vhodné směrovat k podrobnější analýze problému časového posunu, popsaného v sekci 4.4.2, který tvoří velkou překážku při dolování periodických vzorů. Jednou z navrhovaných, ale blíže neprozkoumaných možností, jak tento problém řešit je rozšíření shlukové analýzy o časovou dimenzi, jak je navrženo v sekci 4.5. Odlišná možná cesta budoucího vývoje je studium vlivu meziuživatelských vztahů na periodické chování. Výsledky takové studie by mohly pomoci přizpůsobit dolovací metody a mít tak příznivé účinky na výslednou databázi vzorů a potažmo i na následnou klasifikaci.

Literatura

[1] Al-Naymat, G.; Chawla, S.; Gudmundsson, J.: Dimensionality reduction for long duration and complex spatio-temporal queries. In *Proceedings of the 2007 ACM symposium on Applied computing*, SAC, 2007, s. 393–397.

[2] Ankerst, M.; Breunig, M. M.; Kriegel, H.; aj.: OPTICS: Ordering Points To Identify the Clustering Structure. In *Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, 1999, s. 49–60.

[3] Aung, H.; Tan, K.: Discovery of Evolving Convoys. In *Scientific and Statistical Database Management*, ročník 6187, 2010, s. 196–213.

[4] Bahl, P.; Padmanabhan, V.: RADAR: an in-building RF-based user location and tracking system. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, ročník 2, 2000, s. 775–784.

[5] Blei, D. M.; Ng, A. Y.; Jordan, M. I.: Latent dirichlet allocation. *J. Mach. Learn. Res.*, ročník 3, 2003: s. 993–1022.

[6] Cao, H.; Mamoulis, N.; Cheung, D.: Discovery of Periodic Patterns in Spatiotemporal Sequences. *Knowledge and Data Engineering, IEEE Transactions on*, ročník 19, 2007: s. 453–467.

[7] Eagle, N.; Pentland, A.: Eigenbehaviors: identifying structure in routine. *Behavioral Ecology and Sociobiology*, ročník 63, 2009: s. 1057–1066.

[8] Ester, M.; Kriegel, H.; Sander, J.; aj.: A density-based algorithm for discovering clusters in large spatial databases with noise. 1996, s. 226–231.

[9] Farrahi, K.; Gatica-Perez, D.: Discovering routines from large-scale human locations using probabilistic topic models. *ACM Trans. Intell. Syst. Technol.*, ročník 2, 2011: s. 3:1–3:27.

[10] Fayyad, U.; Piatetsky-Shapiro, G.; Smyth, P.: From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, ročník 17, č. 3, 1996: s. 37–54.

[11] Gaffney, S.; Smyth, P.: Trajectory clustering with mixtures of regression models. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD, 1999, s. 63–72.

[12] Gudmundsson, J.; van Kreveld, M.: Computing longest duration flocks in trajectory data. In *Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems*, GIS, 2006, s. 35–42.

[13] Gudmundsson, J.; van Kreveld, M.; Speckmann, B.: Efficient detection of motion patterns in spatio-temporal data sets. In *Proceedings of the 12th annual ACM international workshop on Geographic information systems*, GIS, 2004, s. 250–257.

[14] Gupta, M.; Chen, Y.: Theory and use of the EM algorithm. *Foundations and Trends in Signal Processing*, ročník 4, 2011: s. 223–296.

[15] Han, J.; Kamber, M.: *Data Mining: Concepts and Techniques. Second Edition.* Elsevier Inc., 2006, iSBN 1-55860-901-3.

[16] Han, J.; Li, Z.; Tang, L.: Mining Moving Object, Trajectory and Traffic Data. In *Database Systems for Advanced Applications*, 2010, s. 485–486.

[17] Hightower, J.: *The Location Stack.* Dizertační práce, University of Washington, 2004.

[18] Hinneburg, A.; Keim, D. A.: An Efficient Approach to Clustering in Large Multimedia Databases with Noise. In *Proceedings KDD*, 1998, s. 58–65.

[19] Huynh, T.; Schiele, B.: Analyzing features for activity recognition. In *Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies*, 2005, s. 159–163.

[20] Jain, A. K.: Data Clustering: 50 Years Beyond K-Means. *Pattern Recognition Letters*, ročník 31, č. 8, 2010: s. 651–666.

[21] Jeung, H.; Shen, H. T.; Zhou, X.: Convoy Queries in Spatio-Temporal Databases. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, 2008, s. 1457–1459.

[22] Jeung, H.; Yiu, M.; Zhou, X.; aj.: Discovery of convoys in trajectory databases. *Proc. VLDB Endow.*, ročník 1, 2008: s. 1068–1080.

[23] Knorr, E.; Ng, R. T.; Tucakov, V.: Distance-based outliers: algorithms and applications. *The VLDB Journal*, ročník 8, 2000: s. 237–253.

[24] Laube, P.; Imfeld, S.: Analyzing Relative Motion within Groups of Trackable Moving Point Objects. In *Geographic Information Science*, Lecture Notes in Computer Science, 2002, s. 132–144.

[25] Laube, P.; van Kreveld, M.; Imfeld, S.: Finding REMO – Detecting Relative Motion Patterns in Geospatial Lifelines. In *Developments in Spatial Data Handling*, 2005, s. 201–215.

[26] Lee, J.; Han, J.; Li, X.: Trajectory Outlier Detection: A Partition-and-Detect Framework. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, 2008, s. 140–149.

[27] Lee, J.; Han, J.; Whang, K.: Trajectory clustering: a partition-and-group framework. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, SIGMOD, 2007, s. 593–604.

[28] Li, Z.; Ding, B.; Han, J.; aj.: Swarm: Mining Relaxed Temporal Moving Object Clusters. *Proc. VLDB Endow.*, ročník 3, 2010: s. 723–734.

[29] Li, Z.; Han, J.; Ji, M.; aj.: MoveMine: Mining moving object data for discovery of animal movement patterns. *ACM Trans. Intell. Syst. Technol.*, ročník 2, 2011: s. 37:1–37:32.

[30] Liao, L.: *Location-based Activity Recognition*. Dizertační práce, University of Washington, 2006.

[31] Liao, L.; Fox, D.; Kautz, H.: Extracting places and activities from gps traces using hierarchical conditional random fields. *The International Journal of Robotics Research*, ročník 26, 2007: s. 119–134.

[32] Miluzzo, E.; Cornelius, C.; Ramaswamy, A.; aj.: Darwin phones: the evolution of sensing and inference on mobile phones. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys, 2010, s. 5–20.

[33] Nanni, M.; Pedreschi, D.: Time-focused clustering of trajectories of moving objects. *Journal of Intelligent Information Systems*, ročník 27, č. 3, 2006: s. 267–289.

[34] Owens, J.; Hunter, A.: Application of the self-organising map to trajectory classification. In *Visual Surveillance, 2000. Proceedings. Third IEEE International Workshop on*, 2000, s. 77–83.

[35] Sheikholeslami, G.; Chatterjee, S.; Zhang, A.: Wavecluster: A multi-resolution clustering approach for very large spatial databases. In *Proceedings of the international conference on very large data bases*, 1998, s. 428–439.

[36] Sinnott, R. W.: Virtues of the Haversine. *Sky and Telescope*, ročník 68, 1984: str. 159.

[37] Wang, L.; Gu, T.; Tao, X.; aj.: Sensor-Based Human Activity Recognition in a Multi-user Scenario. In *Ambient Intelligence*, Lecture Notes in Computer Science, 2009, s. 78–87.

[38] Wang, W.; Yang, J.; Muntz, R.: STING: A statistical information grid approach to spatial data mining. 1997, s. 186–195.

[39] Wikelski, M.; Kays, R.: Movebank: archive, analysis and sharing of animal movement data. [online]. `http://www.movebank.org`, 2012 [cit. 2012-12-26].

[40] Zendulka, J.; Bartík, V.; Lukáš, R.; aj.: *Studijní opora: Získávání znalostí z databází*. FIT VUT v Brně, 2009.

[41] Zhang, T.; Ramakrishnan, R.; Livny, M.: BIRCH: An efficient data clustering method for very large databases. In *Proceedings of the 1996 ACM SIGMOD international conference on Management of data*, 1996, s. 103–114.

[42] Zheng, V.; Yang, Q.: User-dependent aspect model for collaborative activity recognition. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume Three*, IJCAI, 2011, s. 2085–2090.

[43] Zheng, Y.; Li, Q.; Chen, Y.; aj.: Understanding mobility based on GPS data. In *Proceedings of the 10th international conference on Ubiquitous computing*, UbiComp '08, 2008, s. 312–321.

[44] Zheng, Y.; Liu, L.; Wang, L.; aj.: Learning transportation mode from raw gps data for geographic applications on the web. In *Proceedings of the 17th international conference on World Wide Web*, 2008, s. 247–256.

[45] Zheng, Y.; Xie, X.: GeoLife GPS Trajectories [online]. `http://research.microsoft.com/en-us/downloads/b16d359d-d164-469e-9fd4-daa38f2b2e13/`, 2012-08-09 [cit. 2012-12-20].

[46] Zheng, Y.; Zhou, X.: *Computing with Spatial Trajectories*. Springer, 2011, iSBN 978-1-4614-1628-9.